PROGRAM DÖKÜM EKİ

15

- Makine Dilinin Tüm Komutları
- Elmaslar
- Joystick Cursor
- Koca Harfler
- Sentetik Melodi
- Renkli List
- Çizgi Cursor
- Program Kilidi SB

COMMODORE
AYLIK BİLGİSAYAR DERGİSİ

Commodore'un Ücretsiz Ekidir

Commodore

Makine Dilinin Tüm Komutları

AYHAN KALAYLIOĞLU

Burada, COMMODORE 64'ün ana işlem ünitesi MOS 6510'un tüm komutlarının, alfabetik sıraya sokulmuş bir listesini bulacaksınız. Bu listeye, 'bilinmeyen komutlar' da dahil.

Bu bölüm, üç adet tablodan oluşuyor. Birinci tabloda, bilinmeyen komutların listesini; ikinci tabloda, standart komutların listesini, üçüncü tabloda ise, ilk iki tabloda kullanılan kısaltmaların açıklamaları olacak.

Birinci tablo ile bir iki açıklama yapalım:

COMMODORE 64'ün ilk çıktığı günden beri, ana işlem ünitesi MOS 6510'un birçok değişik versiyonu üretildi. Bunlar, standart komut setinde tümüyle kompatibel (uyumlu) olmalarına rağmen, bilinmeyen komutların yürütülmesi sırasında çok değişik davranabiliyorlar. Hangi bilgisayarda, hangisinin tam çalışacağı sorusu ancak deneme ve yanılma yöntemi ile bulunabilir. Birçok monitor programı ile, bu komutlar incelendiğinde, monitor programı bu komutu anlayamadığını belirtmek için yanyana üç adet soru işareti (???) verir. Ocak 1987 sayısında yayınlanan, monitor programı da bu şekilde davranıyor. Fakat, yakında yayınlayacağımız SMON adlı monitor programı, bu tip komutların kullanıldığı programları da incelemenize fırsat verecek.

Gelelim tablolarımıza. Önce birinci tablo:

A11: AND register with $11
X- ve Y-register'i $11 değeri ile AND'lenir. Sonuç, işlem yapılan indeks registerlerinden değeri kullanılarak indeksli adresleme ile belleğe yazılır.

Adresleme Tipleri:

Assembler:	Kod:	Kısaltma:	Byte:
AAX # OP	$8B	IM	2
AAX OP	$87	ZP	2
AAX OP,Y	$97	ZPY	2
AAX OP	$8F	ABS	3
AAX (ZP,X)	$83	(ZP,X)	2

ASR: AND with accu and shift right
Şu işlemler yürütülür:
AND LSR

Adresleme Tipleri:

Assembler:	Kod:	Kısaltma:	Byte:
ASR OP	$6B	IM	2

ARR: AND with accu and rotate right
Bu işlemler yürütülür:
AND ROR

Adresleme Tipleri:

Assembler:	Kod:	Kısaltma:	Byte:
ARR OP	$4B	IM	2

AXS: AND accu and X-register and subtract data
Akü ile X-register'inin AND'lenmesi sonucu elde edilen değerden operand'ın değeri carry'siz çıkartılır ve sonuç X-register'ine yazılır.

Adresleme Tipleri:

Assembler:	Kod:	Kısaltma:	Byte:
AXS OP	$CB	IM	2

DCP: decrement and compare wiht accu
Şu işlemler yürütülür:
DEC CMP

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte:
DCP OP	$C7	ZP	2
DCP OP,X	$D7	ZPX	2
CCP OP	$CF	ABS	3
DCP OP,X	$DF	ABX	3
DCP OP,Y	$DB	ABY	3
DCP (ZP,X)	$C3	(ZP,X)	2
DCP (ZP),Y	$D3	(ZP),Y	2

DOP: double NOP

Bu komutlar yaptıkları işlem bakımından, NOP komutundan farksızdır. Fakat, iki byte sürerler. Burada ikinci byte dikkate alınmaz.

$04, $14, $34, $44, $54, $64, $74, $D4, $F4, $80 $89, $93

ISC: increment and subtract with carry
Şu işlemler yürütülür:
INC SBC

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte:
ISC OP	$E7	ZP	2
ISC OP,X	$F7	ZPX	2
ISC OP	$EF	ABS	3
ISC OP,X	$FF	ABX	3
ISC OP,Y	$FB	ABY	3
ISC (ZP,X)	$E3	(ZP,X)	2
ISC (ZP),Y	$F3	(ZP),Y	2

KIL: Killer codes
Bu komutlar sistemin 'crash' etmesine yolaçar. Böyle durumlarda, RUN/STOP-RESTORE kombinasyonu da bir işe yaramaz. Ya bilgisayarları kapatır açarsınız, ya da resetlersiniz (bir reset switch yardımıyla).

$02, $12, $22, $32, $42, $52, $62, $72, $B2, $D2, $F2

LAR: load accu, and wiht stack-register, transfer result to accu, X-register and stackregister.
Bu işlemler yürütülür:
LDA AND (yığın göstergeci ile) TAX TXS

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte:
LAR OP,Y	$BB	ABY	3

LAX: load to accu and X-register
Şu işlemler yürütülür:
LDA TAX

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte:
LAX OP	$A7	ZP	2
LAX OP,Y	$B7	ZPY	2
LAX OP	$AF	ABS	3
LAX OP,Y	$BF	ABY	3
LAX (ZP,X)	$A3	(ZP,X)	2
LAX (ZP),Y	$B3	(ZP),Y	2

NOP: no operation
Aşağıdaki komutlar, $EA kodlu NOP komutu ile her bakımdan özdeştir.

$1A, $3A, $5A, $7A, $DA, $FA

RLA: rotate left, AND with accu and store accu.
Şu işlemler yürütülür:
ROL AND STA

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte:
RLA OP	$27	ZP	2
RLA CP,X	$37	ZPX	2
RLA OP	$2F	ABS	3
RLA CP,X	$2F	ABX	3
RLA OP,Y	$3B	ABY	3
RLA (ZP,X)	$23	(ZP,X)	2
RLA (ZP),Y	$33	(ZP),Y	2

RRA: rotate right and add with carry.
Bu işlemler yürütülür:
ROR ADC

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte:
RRA OP	$67	ZP	2
RRA OP,X	$77	ZPX	2
RRA OP	$6F	ABS	3
RRA OP,X	$7F	ABX	3
RRA OP,Y	$7B	ABY	3
RRA (ZP,X)	$63	(ZP,X)	2
RRA (ZP),Y	$73	(ZP),Y	2

SLO: shift left and OR with accu.
Şu işlemler yürütülür:
ASL ORA

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte:
SLO OP	$07	ZP	2
SLO OP,X	$17	ZPX	2
SLO OP	$0F	ABS	3
SLO OP,X	$1F	ABX	3
SLO OP,Y	$1B	ABY	3
SLO (ZP,X)	$13	(ZP,X)	2
SLO (ZP),Y	$03	(ZP),Y	2

SRE: shift right and EOR with accu.
Şu işlemler yürütülür:
LSR EOR

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte:
SRE OP	$47	ZP	2
SRE OP,X	$57	ZPX	2
SRE OP	$4F	ABS	3
SRE OP,X	$5F	ABX	3
SRE OP,Y	$5B	ABY	3
SRE (ZP,X)	$43	(ZP,X)	2
SRE (ZP),Y	$53	(ZP),Y	2

TOP: triple NOP
Aşağıdaki komutlar da NOP komutunun bir türüdür. Tek farkları üç byte sürmeleridir. İkinci ve üçüncü byte'ları önemsizdir.

$0C, $1C, $3C, $5C, $7C, $DC, $FC

Geciyoruz ikinci tabloya.

ADC: add with carry

Hafızanın içeriğini aküye, carry'yi de kullanarak topla, sonucu aküye al.

Flaglar: N Z C I D V
+ + + + +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
ADC #OP	$69	IM	2	2
ADC OP	$65	ZP	2	3
ADC OP,X	$75	ZPX	2	4
ADC OP	$6D	ABS	3	4
ADC OP,X	$7D	ABX	3	4
ADC OP,Y	$79	ABY	3	4
ADC (ZP,X)	$61	(ZP,X)	2	6
ADC (ZP),Y	$71	2	5	

AND: AND accu

Hafızanın içeriğini, aküyle AND'le, sonucu aküye al.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
AND #OP	$29	IM	2	2
AND OP	$25	ZP	2	3
AND OP,X	$35	ZPX	2	4
AND OP	$20	ABS	3	4
AND OP,X	$30	ABX	3	4
AND DP,Y	$39	ABY	3	4
AND (ZP,X)	$21	(ZP,X)	2	6
AND (ZP),Y	$31	(ZP),Y	2	5

ASL: arithmetic shift left

Hafızanın içeriğini bir bit pozisyonu sola kaydır.

Flaglar: N Z C I D V
+ + +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
ASL	$0A	Accu	1	2
ASL OP	$06	ZP	2	5
ASL OP,X	$16	ZPX	2	6
ASL OP	$0E	ABS	3	6
ASL OP,X	$1E	ABX	3	7

BCC: branch if carry clear
Carry biti sönükse dallan.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
BBC OP	$90	REL	2	2

BCS: branch if carry set
Carry biti yanıksa dallan.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
BCS OP	$B0	REL	2	2

BEQ: branch if equal (to zero)
Son yapılan işlemin sonucu sıfırsa dallan.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
BEQ OP	$F0	REL	2	2

BIT: test bits
Hafıza ile aküyü AND'le, sonuca göre flag'ları oluştur.

Flaglar: N Z C I D V
+ + +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
BIT OP	$24	ZP	2	3
BIT OP	$2C	ABS	3	4

BMI: branch if minus
Son yapılan işlemin sonunda yedinci bit yanıksa dallan.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
BMI OP	$30	REL	2	2

BNE: branch if not equal (to zero)
Son yapılan işlemin sonucu sıfır değilse dallan.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
BNE OP	$D0	REL	2	2

BRK: break
Programın durması ve break vektörü üzerinden atlama

Flaglar: N Z C I D V B
+

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
BRK	$00	—	1	7

BVC: branch if overflow clear
Son yapılan işlemin sonunda taşma yoksa dallan.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
BVC OP	$50	REL	2	2

BVS: branch if overflow set
Son yapılan işlemin sonunda taşma varsa dallan

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte:	Takt
BVS OP	$70	REL	2	2

CLC: clear carry
Carry bitini söndür.

Flaglar: N Z C I D V
+

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
CLC	$18	—	1	2

CLD: clear decimal mode
Ondalık mod bitini söndür.

Flaglar: N Z C I D V
+

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
CLD	$D8	—	1	2

CLI: clear interrupt disable bit
Interrupt'a izin vermeyen biti söndür.

Flaglar: N Z C I D V
+

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
CLI	$50	—	1	2

CLV: clear overflow flag
Taşma flag'ını söndür.

Flaglar: N Z C I D V
+

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
CLV	$B8	—	1	2

CMP: compare with accu
Hafızanın içeriğini, aküyle karşılaştır.

Flaglar: N Z C I D V
+ + +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
CMP#OP	$C9	IM	2	2
CMP OP	$C5	ZP	2	3
CMP OP,X	$D5	ZPX	2	4
CMP OP	$CD	ABS	3	4
CMP OP,X	$DD	ABX	3	4
CMP DP,Y	$D9	ABY	3	4
CMP (ZP,X)	$C1	(ZP,X)	2	6
CMP (ZP),Y	$D1	(ZP),Y	2	5

CPX: compare with X-register
Hafızanın içeriğini, X-registeriyle karşılaştır.

Flaglar: N Z C I D V
+ + +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
CPX#OP	$E0	IM	2	2
CPX OP	$E4	ZP	2	3
CPX OP	$EC	ABS	3	4

CPY: compare with Y-register
Hafızanın içeriğini, Y-registeriyle karşılaştır.

Flaglar: N Z C I D V
+ + +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
CPY OP	$C0	IM	2	2
CPY OP	$C4	ZP	2	3
CPY OP	$CC	ABS	3	4

DEC: decrement
Hafızanın içeriğini bir eksilt.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
DEC OP	$C6	ZP	2	5
DEC OP,X	$D6	ZPX	2	6
DEC OP	$CE	ABS	3	6
DEC OP,X	$DE	ABX	3	7

DEX: decrement X-register
X-registerinin içeriğini bir eksilt.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
DEX	$CA	—	1	2

DEY: decrement Y-register
Y-registerinin içeriğini bir eksilt.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
DEY	$88	—	1	2

EOR: exclusive-or with accu
Hatızanın içeriğini, aküyle exclusive-or işlemine sok.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
EOR #OP	$49	IM	2.	2
EOR OP	$45	ZP	2	3
EOR OP,X	$55	ZPX	2	4
EOR OP	$40	ABS	3	4
EOR OP,X	$50	ABX	3	4
EOR OP,Y	$59	ABY	3	4
EOR (ZP,X)	$41	(ZP,X)	2	6
EOR (ZP),Y	$51	(ZP),Y	2	5

INC: increment
Hafızanın içeriğini bir artır.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
INC OP	$E6	ZP	2	5
INC OP,X	$F6	ZPX	2	6
INC OP	$EE	ABS	3	6
INC OP,X	$FE	ABX	3	7

INX: increment X-register
X-registerinin içeriğini bir artır.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
INX	$E8	—	1	2

INY: increment Y-register
Y-registerinin içeriğini bir artır.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
INY	$C8	—	1	2

JUM: Jump
Adrese sıçra.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
JMP OP	$4C	ABS	3	3
JMP (OP)	$6C	IND	3	5

JSR: jump to subroutine
Altprogramı çağır.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
JSR OP	$20	ABS	3	6

LDA: load accu
Aküye, hafızanın içeriğini yükle.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
LDA OP	$A9	IM	2	2
LDA OP	$A5	ZP	2	3
LDA OP,X	$B5	ZPX	2	4
LDA OP	$AD	ABS	3	4
LDA OP,X	$81	ABX	3	4
LDA OP,Y	$89	ABY	3	4
LDA (ZP,X)	$A1	(ZP,X)	2	6
LDA (ZP),Y	$B1	(ZP),Y	2	5

LDX: load X-register
X-registerine hafızanın içeriğini yükle.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
LDX OP	$A2	IM	2	2
LDX OP	$A6	ZP	2	3
LDX OP,Y	$B6	ZPY	2	4
LDX OP	$AE	ABS	3	4
LDX OP,Y	$BE	ABY	3	4

LDY: load Y-register
Y-registerine, hafızanın içeriğini yükler.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
LDY OP	$A0	IM	2	2
LDY OP	$A4	ZP	2	3
LDY OP,X	$B4	ZPY	2	4
LDY OP	$AC	ABS	3	4
LDY OP,X	$BC	ABY	3	4

LSR: logical shift right
Hafızanın içeriğini bir bit pozisyonu sağa kaydır. Yedinci bite '0' yüklenir.

Flaglar: N Z C I D V
+ + +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
LSR	$4A	Accu	1	2
LSR OP	$46	ZP	2	5
LSR OP,X	$56	ZPX	2	6
LSR OP	$4E	ABS	3	6
LSR OP,X	$5E	ABX	3	7

NOP: no operation
Hiçbir işlem yapılmaz.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
NOP	$EA	—	1	2

ORA: OR with accu
Aküyle, hafızanın içeriğini OR'la, sonucu aküye yükle.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
ORA #OP	$09	IM	2	2
ORA OP	$05	ZP	2	3
ORA OP,X	$15	ZPX	2	4
ORA OP	$0D	ABS	3	4
ORA OP,X	$1D	ABX	3	4
ORA OP,Y	$19	ABY	3	A
ORA (ZP,X)	$01	(ZP,X)	2	6
ORA (ZP),Y	$11	(ZP).Y	2	5

PHA: push accu
Akümülatörün içeriğini yığında sakla.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
PHA	$48	—	1	3

PHP: push processor-status
Durum registerlerinin içeriğini yığında sakla.

Praglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
PHP	$08	—	1	3

PHA: pull accu
Akümülatörün içeriğini yığından al.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
PLP	$28	—	1	4

ROL: rotate left
Hafızanın içeriğini bir bit pozisyonu sola döndür.

Flaglar: N Z C I D V
+ + +

Commodore

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
ROL	$2A	Accu	1	2
ROL OP	$26	ZP	2	5
ROL OP,X	$36	ZPX	2	6
ROL OP	$2E	ABS	3	6
ROL OP,X	$3E	ABX	3	7

ROR: rotate right
Hafızanın içeriğini bir bit pozisyonu sağa döndür.

Flaglar: N Z C I D V
+ + +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
ROR	$6A	Accu	1	2
ROR OP	$66	ZP	2	5
ROR OP,X	$76	ZPX	2	6
ROR OP	$6E	ABS	3	6
ROR OP,X	$7E	ABX	3	7

RTI: return from interrupt
Interrupt işlemi bittikten sonra, kesilen programa kaldığı yerden devam et.

Flaglar: N Z C I D V
İnterrupt'tan önceki gibi

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
RTI	$40	—	1	6

RTS: return from subroutine
Altprogramdan geri dön.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
RTS	$60	—	1	6

SBC: subtract with carry
Aküden, hafızanın içeriğini, carry'yi de kullanarak çıkar.

Flaglar: N Z C I D V
+ + + +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
SBC# OP	$E9	IM	2	2
SBC OP	$E5	ZP	2	3
SBC OP,X	$F5	ZPX	2	4
SBC OP	$ED	ABS	3	4
SBC OP,X	$FD	ABX	3	4
SBC OP,Y	$F9	ABY	3	4
SBC (ZP,X)	$E1	(ZP,X)	2	6
SBC (ZP),Y	$F1	(ZP),Y	2	5

SEC: set carry flag
Carry bitini yak.

Flaglar: N Z C I D V
+

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
SEC	$38	—	1	2

SED: set decimal mode
Desimal mode geçme flag'ını yak.

Flaglar: N 2 C I D V
+

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
SED	$F8	—	1	2

SEI: set interrupt disablebit
Interrupt'a izin vermeyen biti yak.

Flaglar: N Z C I D V
+

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
SEI	$78	—	1	2

STA: store accu
Aküyü, hafızaya sakla.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
STA OP	$85	ZP	2	3
STA OP,X	$95	ZPX	2	4
STA OP	$80	ABS	3	4
STA OP,X	$90	ABX	3	5
STA OP,Y	$99	ABY	3	5
STA (ZP,X)	$81	(ZP,X)	2	6
STA (ZP),Y	$91	(ZP),Y	2	6

STX: store X-register
X-registerini hafızaya sakla.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
STX OP	$86	ZP	2	3
STX OP,Y	$96	ZPY	2	4
STX OP	$8E	ABS	3	4

STY: store Y-register
Y-registerini, hafızaya sakla.

Flaglar: N Z C I O V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
STY OP	$84	ZP	2	3
STY OP,X	$94	ZPY	2	4
STY OP	$8C	ABS	3	4

TAX: transfer accu to X-register
Aküyü, X-registerine sakla.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
TAX	$AA	—	1	2

TAY: transfer accu to Y-register
Aküyü, Y-registerine sakla

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
TAY	$A8	—	1	2

TSX: transfer stackpointer to X-register
Yığın göstergecini, X-registerine sakla.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
TSX	$BA	—	1	2

TXA: transfer X-register to accu
X-registerini, aküye sakla.

Flaglar: N Z C I D V
+ +

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
TXA	$8A	—	1	2

TXS: transfer X-register to stackpointer
X-registerini, yığın göstergecine kopyala.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
TXS	$9A	—	1	2

TYA: transfer Y-register to accu
Y-registerini, aküye sakla.

Flaglar: N Z C I D V
hiçbiri

Adresleme Tipleri

Assembler:	Kod:	Kısaltma:	Byte	Takt
TYA	$98	—	1	2

Ve son olarak üçüncü tablo:

ABS: absolut
Operand dört basamaklı, hekza-

desimal bir sayıdır.
Örnek:

LDA $C000
$C000 adresli hafızanın içeriği aküye yüklenir.

ABX: absolut X-indeksli
Operand dört basamaklı, hekzadesimal bir sayıdır. X-registerinin içeriği, operanda toplanır ve böylece aktif adres elde edilir. İşlem, bu adresle yürütülür.
Örnek:

LDX #$10
LDA $C000,X
$C010 ($C000+$10) adresli hafızanın içeriği aküye yüklenir.

ABY: absolut Y-indeksli
Operand dört basamaklı, hekzadesimal bir sayıdır. Y-registerinin içeriği, operanda toplanır ve böylece aktif adres elde edilir. İşlem, bu adresle yürütülür.
Örnek:
LDY #$10
LDA $C000,Y
$C010 ($C000+$10) adresli hafızanın içeriği aküye yüklenir.
Byte: İlk iki tabloda kullanılan bu kısaltmanın anlamı, tüm bir komutun, operandlarıyla beraber uzunluğudur.
Flaglar: Durum registerinin her bir biti.
N: negative flag. Son yapılan işlemde, yedinci bitin yanık olup olmadığını gösterir.
Z: zero flag. Son yapılan işlemin sonucunun sıfır olup olmadığını gösterir.
C: carry flag. Son yapılan işlemdeki carry durumunu gösterir.
I: interrupt disable flag. Bu flag yanık olduğu zaman interrupt'lar engellenir.
D: decimal mode flag. Bu flag yakıldığında, ana işlem ünitesi, kısmen desimal (ondalık) modda işlem yapar. Örneğin: bu modda $01 ile $09'u toplarsak $0A değil $10 elde ederiz. Böylece ikili-onlu ve onlu-ikili dönüşümlerinden kurtuluruz.
V: overflow flag. Son yapılan işlemde taşıma (overflow) olup olmadığını gösterir.
IM: immediate
Bu adresleme tipinde, operand bir byte uzunluğundadır ve bu operand doğrudan değeri ile kullanılır.
Örnek:
LDA #$00
$00 değeri doğrudan aküye yüklenir.
OP: operand
Her adresleme tipinin gerektirdiği ve komutla beraber yürüyen komutun uzantıları.
ZP: zeropage (sıfırıncı sayfa)
Adres, iki yerine bir byte'la gösterilir. Diğer byte olarak $00 kullanılır. Bu yöntemle, yalnızca hafızanın ilk 256 byte'ını adresleyebiliriz. ($0000'dan $00FF'e kadar)
Örnek:

LDA $2B
$002B adresli hafızanın içeriği aküye yüklenir.
ZPX: zeropage X-indeksli
Adres için belirtilen byte'a X-register'i toplanır. Sonucun sadece sekiz biti kullanılır. Yukarıdaki şekilde onaltı bite genişletilir.
Örnek:

LDX #$30
LDA $F0,X
Yüklenen adres $0120 değil, $0020'dir. Bu adreslemenin sonunda sıfırıncı sayfanın dışına çıkamayız.

(ZP,X): indersli indirekt
Operand byte' ile X-registeri toplanır. Sonuçta sıfırıncı sayfada, esas istediğimiz adresi belirten göstergecin adresini eide ederiz.
Örnek:
$20 adresli hafıza $00 içersin.
$21 adresli hafıza $CD içersin.
LDX #$0E
LDA ($12,X)
$0020 ($0012+$000E) ve $0021 hafızaları $C000 adresini gösterir. Aküye, bu adresin içeriği yüklenir.
(ZP),Y: indirekt indeksli
Operand byte'ı, sıfırıncı sayfada bir göstergecin sonrası oluşturur. Bu göstergece, Y-registerinin içeriği de toplanınca esas çalışma süresi elde edilir.

Örnek:
$20 adresli hafıza $00 içersin.
$21 adresli hafıza $C0 içersin.
LDY #$10
LDA ($20),Y

$C010 ($C000+$0010) adresli hafızanın içeriği aküye yüklenir.
Takt: Komutun harcadığı zaman COMMODORE 64, aşağı yukarı 1 MHz civarında bir frekansla çalışır. Buna göre her bir takt bir mikrosaniye (saniyenin milyonda biri) sürer. Bu hesapla herhangi bir komutun ne kadar zaman süreceğini bulabiliriz.

Örnek:
RTS işlemi 6 mikrosaniye, LDA $00 işlemi de 2mikrosaniye sürer. Yani saniyede LDA $00 benzeri işlemlerden 500000 adet yürütülebilir.

Digitizer Yazısına Ait Program:

Speech Basic

```
 PROGRAM  : SPEECH BASIC     0801 17E1
--------------------------------------

0801 : 1E 08 C2 07 9E 28 32 30   09
0809 : 38 30 29 20 53 50 45 45   FF
0811 : 43 48 20 42 41 53 49 43   23
0819 : 20 32 2E 37 00 00 00 4C   5D
0821 : C0 09 A6 7A A0 04 84 0F   B9
0829 : BD 00 02 10 07 C9 FF F0   0A
0831 : 3F E8 D0 F4 C9 20 F0 38   83
0839 : 85 08 C9 22 F0 56 24 0F   E9
0841 : 70 2E C9 3F D0 04 A9 99   2A
0849 : D0 26 C9 30 90 04 C9 3C   6D
0851 : 90 1E 34 71 A0 4C 84 0B   D4
0859 : A0 FF 86 7A CA C8 E8 BD   FC
0861 : 00 02 38 F9 60 0A F0 F5   B6
0869 ,: C9 80 D0 2F 05 0B A4 71   AA
0871 : C8 E8 99 FB 01 C9 00 F0   D3
0879 : 38 38 E9 3A F0 04 C9 49   78
0881 : D0 02 85 0F 38 E9 55 D0   5F
0889 : 9F 85 08 BD 00 02 F0 E0   3A
0891 : C5 08 F0 DC C8 99 FB 01   7D
0899 : E8 D0 F0 A6 7A E6 0B C8   97
08A1 : B9 5F 0A 10 FA B9 60 0A   A1
08A9 : D0 B5 F0 0F BD 00 02 10   76
08B1 : BD 99 FD 01 C6 7B A9 FF   C9
08B9 : 85 7A 60 A0 00 84 0B B9   6B
08C1 : 60 0A D0 02 C8 E8 BD 00   65
08C9 : 02 38 F9 9E A0 F0 F5 C9   36
08D1 : 80 D0 04 05 0B D0 97 A6   3E
08D9 : 7A E6 0B C8 B9 9D A0 10   CD
08E1 : FA B9 9E A0 D0 E0 F0 C4   D5
08E9 : 48 AD 8D 02 C9 01 F0 F9   08
08F1 : 68 10 0F 24 0F 30 0B C9   DC
08F9 : FF F0 07 C9 CC B0 06 4C   6E
0901 : 24 A7 4C F3 A6 38 E9 CB   F6
0909 : AA 84 49 A0 FF CA F0 08   86
0911 : C8 B9 60 0A 10 FA 30 F5   35
0919 : C8 B9 60 0A 30 05 20 47   51
0921 : AB D0 F5 4C EF A6 20 73   D7
0929 : 00 20 30 09 4C AE A7 C9   03
0931 : CC 90 04 C9 E4 90 06 20   AB
0939 : 79 00 4C ED A7 38 E9 CC   01
0941 : 0A AA BD D3 0A 48 BD D2   0A
0949 : 0A 48 4C 73 00 A9 00 85   51
0951 : 0D 20 73 00 C9 24 F0 0A   E1
0959 : C9 25 F0 41 20 79 00 4C   80
0961 : 8D AE 20 94 09 20 73 00   3F
0969 : 90 0B C9 41 90 1F C9 47   D1
0971 : B0 1B 38 E9 07 38 E9 30   34
0979 : 48 A5 61 F0 07 18 69 04   E9
0981 : B0 0E 85 61 68 F0 DE 20   90
0989 : 7E BD 4C 66 09 4C 79 00   3F
0991 : 4C 7E B9 A9 00 A2 0A 95   23
0999 : 5D CA 10 FB 60 20 94 09   4A
09A1 : 20 73 09 C9 32 B0 E6 C9   8C
```

```
09A9 : 30 90 E2 E3 30 48 A5 61  B6
09B1 : F0 04 E6 61 F0 DA 68 F0  F2
09B9 : E7 20 7E BD 4C A1 09 A9  51
09C1 : 01 A0 18 85 2B 84 2C A9  A4
09C9 : 00 8D 00 18 20 D3 09 4C  F0
09D1 : 1F 0A A9 23 8D 04 03 A9  1C
09D9 : 08 8D 05 03 A9 E9 8D 06  76
09E1 : 03 A9 08 8D 07 03 A9 27  EA
09E9 : 8D 08 03 A9 09 8D 09 03  38
09F1 : A9 4E 8D 0A 03 A9 09 8D  23
09F9 : 0B 03 A9 F7 8D 8F 02 A9  A0
0A01 : 0C 8D 90 02 A9 51 8D 18  C4
0A09 : 03 A9 0D 8D 19 03 A9 61  E9
0A11 : 8D 16 03 A9 0D 8D 17 03  3F
0A19 : A9 0F 8D 89 02 60 A5 2B  EE
0A21 : A4 2C 20 08 A4 A9 37 A0  9A
0A29 : 0A 20 1E AB A9 98 A0 E4  EC
0A31 : 20 2D E4 4C 9D E3 93 08  02
0A39 : 0E 0D 20 20 20 20 2A 2A  DA
0A41 : 2A 2A 20 C3 36 34 20 D3  2E
0A49 : 50 45 45 43 48 20 D3 59  7D
0A51 : 53 54 45 4D 20 56 32 2E  A3
0A59 : 37 20 2A 2A 2A 2A 00 52  09
0A61 : 45 53 45 D4 42 41 53 49  4A
0A69 : C3 48 45 4C D0 4B 45 D9  5B
0A71 : 48 49 4D 45 CD 44 49 53  25
0A79 : CB 44 49 D2 42 4C 4F 41  59
0A81 : C4 42 53 41 56 C5 4D 41  AF
0A89 : D0 4D 45 CD 50 41 55 53  16
0A91 : C5 42 4C 4F 43 CB 48 45  B3
0A99 : 41 D2 52 45 43 4F 52 C4  02
0AA1 : 50 4C 41 D9 56 4F 4C 44  3C
0AA9 : 45 C6 43 4F 4C 44 45 C6  96
0AB1 : 48 45 D8 44 45 DA 53 43  59
0AB9 : 52 45 45 CE 45 58 45 C3  8D
0AC1 : 4D 4F CE 0F 46 52 4F CD  35
0AC9 : 53 50 45 45 C4 4F 46 C6  AC
0AD1 : 00 01 0B 27 0B 5D 0B A5  0D
0AD9 : 0B 71 0D 97 0D E2 0D 4F  8E
0AE1 : 0E 6E 0F CA 10 E5 0F 80  B1
0AE9 : 10 A4 11 87 12 DC 13 76  C2
0AF1 : 13 55 14 72 14 8B 14 91  13
0AF9 : 14 97 14 BE 14 55 15 F4  E0
0B01 : 15 A9 04 8D 88 02 20 5B  6D
0B09 : FF 20 15 FD 20 53 E4 20  8E
0B11 : D3 09 A9 08 20 D2 FF A9  C0
0B19 : 0E 20 D2 FF A9 00 A2 18  41
0B21 : 9D 00 D4 CA 10 FA 60 24  EF
0B29 : 9D 10 19 A9 4F A0 0B 20  B0
0B31 : 1E AB 20 CF FF 29 7F C9  02
0B39 : 59 08 20 CF FF C9 0D D0  BC
0B41 : F9 28 D0 09 20 15 FD 20  86
0B49 : 53 E4 20 18 E5 60 41 52  24
0B51 : 45 20 59 4F 55 20 53 55  35
0B59 : 52 45 3F 20 00 D0 0B A9  28
0B61 : 60 85 A6 A9 0A 85 A7 4C  67
0B69 : 76 0B A9 9E 85 A6 A9 A0  18
0B71 : 85 A7 20 73 00 A0 00 B1  A9
0B79 : A6 F0 29 48 29 7F 20 D2  9F
0B81 : FF C8 68 10 F2 84 A8 38  67
0B89 : 20 F0 FF 98 38 E9 0A B0  91
0B91 : FC 49 FF 69 01 AA E8 CA  FD
0B99 : D0 05 A4 A8 4C 78 0B 20  1F
0BA1 : 3B AB D0 F3 60 D0 2E A9  FD
0BA9 : 00 48 20 0E 0C A2 00 20  AD
0BB1 : 23 0C 68 48 AA E8 A9 00  96
0BB9 : 20 CD BD A2 05 20 23 0C  79
0BC1 : A5 A6 A4 A7 20 34 0C A2  F0
0BC9 : 08 20 23 0C 68 AA E8 8A  C0
0BD1 : C9 08 D0 D5 60 20 9E B7  7E
0BD9 : CA 10 05 A2 0E 4C 37 A4  AA
0BE1 : E0 08 B0 F7 8A 20 0E 0C  EA
0BE9 : 20 FD AE 20 9E AD 20 A3  D7
0BF1 : B6 C9 10 90 05 A2 17 4C  FC
0BF9 : 37 A4 A8 A9 00 F0 08 B1  ED
0C01 : 22 C9 5F D0 02 A9 0D 91  BE
0C09 : A6 88 10 F3 60 0A 0A 0A  08
0C11 : 0A 18 69 6F 85 A6 A9 0C  BC
0C19 : 69 00 85 A7 60 24 9D 30  D7
0C21 : 01 60 BD 51 0C F0 06 20  8C
0C29 : D2 FF E8 D0 F5 60 A9 0D  72
0C31 : 4C D2 FF 85 A6 84 A7 A0  05
0C39 : 00 B1 A6 F0 12 48 29 7F  E1
0C41 : C9 0D D0 04 68 A9 5F 24  DF
0C49 : 68 20 D2 FF C8 D0 EA 60  F5
0C51 : 4B 45 59 20 00 2C 22 00  83
0C59 : 22 0D 00 42 4C 4F 43 4B  2D
0C61 : 20 00 20 46 52 4F 4D 20  67
0C69 : 00 20 54 4F 20 00 93 52  6D
0C71 : 55 4E 0D 00 20 20 20 20  F5
0C79 : 20 20 20 20 20 20 93 4C  9F
0C81 : 49 53 54 0D 00 20 20 20  EC
0C89 : 20 20 20 20 20 20 50 4C  A2
0C91 : 41 59 20 00 20 20 20 20  4B
0C99 : 20 20 20 20 20 20 48 45  34
0CA1 : 41 52 3A 52 45 43 4F 52  35
0CA9 : 44 20 00 20 20 20 93 48  E3
0CB1 : 45 4C 50 0D 4B 45 59 0D  30
0CB9 : 00 20 20 20 20 20 42 4C  7A
0CC1 : 4F 43 4B 20 00 20 20 20  4B
0CC9 : 20 20 20 20 20 20 93 44  DF
0CD1 : 49 52 0D 00 20 20 20 20  4B
0CD9 : 20 20 20 20 20 20 44 49  BC
0CE1 : 53 4B 0D 00 20 20 20 20  E1
0CE9 : 20 20 20 20 20 20 00 02  2D
0CF1 : 04 06 01 03 05 07 20 48  33
0CF9 : EB A4 C6 88 38 B9 77 02  2C
0D01 : E9 85 B0 01 60 C9 08 B0  CF
0D09 : FB A8 A5 A6 48 A5 A7 48  77
0D11 : B9 EF 0C 20 0E 0C A0 00  8D
0D19 : A6 C6 CA B1 A6 F0 09 9D  5C
0D21 : 77 02 C8 E8 E0 0F D0 F3  9A
0D29 : 86 C6 68 85 A7 68 85 A6  FE
0D31 : 60 18 AD B8 10 6D B9 10  93
0D39 : D0 02 90 11 38 AD B8 10  45
0D41 : E9 01 8D B8 10 AD B9 10  9B
0D49 : E9 00 8D B9 10 4C 31 EA  CB
0D51 : 48 8A 48 98 48 20 BC F6  6A
0D59 : 20 E1 FF F0 03 4C 72 FE  E2
0D61 : A9 7F 20 C3 FF 20 15 FD  9C
0D69 : 20 A3 FD 20 02 0B 6C 02  0C
0D71 : A0 20 2F 0F C9 A0 F0 05  3E
```

Commodore

```
0D79 : 90 07 4C DC 0B E0 00 D0  35
0D81 : F9 C5 2E 90 F5 F0 03 4C  86
0D89 : 8F 0D E4 2D 90 EC 86 37  77
0D91 : 85 38 A9 00 4C 5E A6 A9  42
0D99 : 00 20 BD FF A9 7F A2 08  4A
0DA1 : A0 0F 20 BA FF 20 CB 0D  72
0DA9 : 20 C0 FF A5 B7 D0 12 A2  6D
0DB1 : 7F 20 C6 FF 20 E4 FF C9  AF
0DB9 : 0D F0 06 20 D2 FF 4C B5  8E
0DC1 : 0D 20 CC FF A9 7F 20 C3  B0
0DC9 : FF 60 20 06 E2 20 57 E2  14
0DD1 : 20 06 E2 20 00 E2 E0 08  5C
0DD9 : B0 05 A2 09 4C 37 A4 86  F4
0DE1 : BA 60 A9 01 A2 A6 A0 00  38
0DE9 : 20 BD FF A9 7F A2 08 A0  8B
0DF1 : 00 20 BA FF A9 24 85 A6  CF
0DF9 : 20 CB 0D A9 00 85 90 20  26
0E01 : C0 FF A2 7F 20 C6 FF 20  D2
0E09 : 3F 0E 20 3F 0E 20 3F 0E  3A
0E11 : 20 CD BD A9 20 20 D2 FF  0B
0E19 : 20 E4 FF A6 90 D0 16 20  A8
0E21 : D2 FF D0 F4 AD 8D 02 C9  A8
0E29 : 01 F0 F9 C9 02 F0 06 20  5A
0E31 : 2F 0C 4C 0B 0E 20 CC FF  F0
0E39 : A9 7F 20 C3 FF 60 20 E4  70
0E41 : FF AA 20 E4 FF A4 90 F0  83
0E49 : 05 68 68 4C 36 0E 60 A9  CF
0E51 : 00 85 0A 85 90 20 BD FF  48
0E59 : A9 7F A2 08 A0 00 20 BA  6B
0E61 : FF 20 CB 0D A5 B7 D0 05  6A
0E69 : A2 08 4C 37 A4 20 AF F5  FF
0E71 : 20 C0 FF A2 7F 20 C6 FF  5A
0E79 : 20 E4 FF 48 A5 90 F0 0D  D1
0E81 : 20 CC FF A9 7F 20 C3 FF  44
0E89 : A2 04 4C 37 A4 20 D2 F5  AA
0E91 : 68 20 42 0E 18 20 45 0F  11
0E99 : 20 64 0F A5 AE 8D 5B 16  55
0EA1 : A5 AF 8D 5C 16 20 E4 FF  03
0EA9 : 78 A0 34 84 01 A0 00 91  47
0EB1 : AE A0 37 84 01 58 E6 AE  DA
0EB9 : D0 02 E6 AF A4 90 D0 0C  64
0EC1 : A5 AF C5 A9 D0 DF A5 AE  E4
0EC9 : C5 A8 D0 D9 20 CC FF A9  0D
0ED1 : 7F 20 C3 FF A5 AE 8D 5D  12
0ED9 : 16 A5 AF 8D 5E 16 A2 13  B3
0EE1 : 20 1E 0C A6 AE A5 AF 20  FF
0EE9 : EC 0E 60 24 9D 30 01 60  99
0EF1 : 2C 8B 14 30 03 4C CD BD  33
0EF9 : A8 8A 48 98 48 A9 24 20  AE
0F01 : D2 FF 68 F0 03 20 0A 0F  82
0F09 : 68 48 4A 4A 4A 4A 20 15  13
0F11 : 0F 68 29 0F AA BD 1C 0F  A8
0F19 : 4C D2 FF 30 31 32 33 34  AE
0F21 : 35 36 37 38 39 41 42 43  73
0F29 : 44 45 46 20 FD AE 20 8A  90
0F31 : AD 20 F7 B7 A6 14 A5 15  AF
0F39 : C9 FF D0 07 E0 F9 90 03  3D
0F41 : 4C DC 0B 60 08 20 0E 12  A8
0F49 : A9 12 28 69 00 AA 20 1E  45
0F51 : 0C AE 84 12 AD 85 12 86  74
0F59 : AE 85 AF 20 E4 0E A6 AE  70
0F61 : A5 AF 60 AE 86 12 AD 87  8A
0F69 : 12 86 A8 85 A9 60 A9 00  DD
0F71 : 20 BD FF A9 7F A2 08 A0  13
0F79 : 01 20 BA FF 20 CB 0D A5  19
0F81 : B7 D0 03 4C 69 0E A9 00  98
0F89 : 85 90 20 C0 FF 20 8F F6  A4
0F91 : 20 2F 0C 38 A6 37 A5 38  7E
0F99 : 20 45 0F 20 64 0F A2 7F  6C
0FA1 : 20 C9 FF A5 AE 20 D2 FF  92
0FA9 : A5 AF 20 D2 FF 78 A0 34  37
0FB1 : 84 01 A0 00 B1 AE A0 37  5F
0FB9 : 84 01 58 20 D2 FF A5 90  BD
0FC1 : F0 06 20 CC FF 4C 36 0E  AD
0FC9 : E6 AE D0 02 E6 AF A5 AF  5D
0FD1 : C5 A9 D0 D9 A5 AE C5 A8  12
0FD9 : D0 D3 20 CC FF A9 7F 20  C0
0FE1 : C3 FF 4C DF 0E A9 3B 85  D9
0FE9 : AE A9 10 85 AF A0 00 B1  84
0FF1 : AE 48 C8 B1 AE A8 68 C0  7F
0FF9 : FF F0 2A 20 1E AB A0 02  C5
1001 : 20 27 10 A9 0D 20 67 11  80
1009 : A2 19 20 23 0C A0 04 20  BA
1011 : 27 10 20 2F 0C 18 A5 AE  A4
1019 : 69 06 85 AE A5 AF 69 00  3A
1021 : 85 AF 4C EE 0F 60 B1 AE  87
1029 : 85 A6 C8 B1 AE 85 A7 A0  61
1031 : 00 B1 A6 AA C8 B1 A6 4C  56
1039 : F1 0E 55 10 2B 00 2D 00  F0
1041 : 5E 10 37 00 53 10 67 10  E9
1049 : 79 10 7B 10 70 10 7D 10  49
1051 : 7F 10 F8 FF 42 41 53 49  34
1059 : 43 20 3A 20 00 53 4F 55  C1
1061 : 4E 44 20 3A 20 00 4B 45  DA
1069 : 59 53 20 20 3A 20 00 42  A1
1071 : 4C 4F 43 4B 53 3A 20 00  27
1079 : 6F 0C EF 0C 5B 16 DB 17  70
1081 : 20 79 00 F0 34 20 2F 0F  9B
1089 : 8E B8 10 8D B9 10 78 AD  82
1091 : 14 03 48 AD 15 03 48 A9  CC
1099 : 32 8D 14 03 A9 0D 8D 15  5B
10A1 : 03 58 AD B8 10 D0 FB AD  26
10A9 : B9 10 D0 F6 78 68 8D 15  A8
10B1 : 03 68 8D 14 03 58 60 00  43
10B9 : 00 EA AD 00 DC CD 00 DC  90
10C1 : F0 FB EA EA EA EA EA EA  4E
10C9 : EA 60 08 A9 00 8D 65 11  3F
10D1 : A9 20 8D 66 11 28 F0 23  21
10D9 : 20 79 00 20 72 11 8D 65  6B
10E1 : 11 8E 66 11 20 79 00 F0  A5
10E9 : 17 C9 AB F0 01 60 20 73  68
10F1 : 00 D0 07 A9 20 8D 66 11  7B
10F9 : D0 06 20 72 11 8E 66 11  64
1101 : AD 65 11 48 20 92 11 A2  CE
1109 : 0B 20 23 0C 68 48 AA A9  35
1111 : 00 20 CD BD A9 08 20 67  76
1119 : 11 A2 12 20 23 0C A0 00  19
1121 : 20 88 11 A9 13 20 67 11  F1
1129 : A2 19 20 23 0C A0 02 20  D2
1131 : 88 11 A9 20 20 D2 FF A9  9C
1139 : 1D 20 67 11 A9 22 20 D2  34
1141 : FF A4 A8 A2 08 B9 DB 16  FB
```

Commodore

```
1149 : 20 D2 FF C8 CA D0 F6 A9  4E
1151 : 22 20 D2 FF 20 2F 0C 68  B4
1159 : AA 20 2C A8 E8 8A CD 66  1A
1161 : 11 D0 A0 60 00 00 48 38  A0
1169 : 20 F0 FF 68 A8 18 4C F0  6D
1171 : FF 20 6B A9 A5 15 F0 03  5D
1179 : 4C DC 0B A5 14 C9 20 90  DC
1181 : 03 4C DC 0B AA E8 60 B1  1A
1189 : AE AA C8 B1 AE 20 F1 0E  C4
1191 : 60 0A 0A 48 18 69 5B 85  C7
1199 : AE A9 16 69 00 85 AF 68  8A
11A1 : 0A 85 A8 60 D0 03 4C CB  92
11A9 : 10 20 9E B7 E0 00 F0 04  42
11B1 : E0 20 90 03 4C DC 0B 8A  13
11B9 : 20 92 11 A5 AE 85 A6 A5  18
11C1 : AF 85 A7 A6 37 A5 38 20  B3
11C9 : 0E 12 A0 03 B9 84 12 91  94
11D1 : A6 88 10 F8 20 79 00 F0  8E
11D9 : 33 A4 A8 A2 08 A9 20 99  5E
11E1 : DB 16 C8 CA D0 F9 20 9E  ED
11E9 : AD 20 A3 B6 C9 00 D0 01  48
11F1 : 60 C9 09 90 03 4C F6 0B  0F
11F9 : 86 AA 84 AB A8 88 98 18  CC
1201 : 65 A8 AA B1 AA 9D DB 16  CE
1209 : CA 88 10 F7 60 8E 84 12  CB
1211 : 8D 35 12 A2 F8 A9 FF 8E  34
1219 : 86 12 8D 87 12 20 79 00  05
1221 : C9 D8 D0 1F 20 73 00 20  4C
1229 : 9E B7 E0 20 B0 86 A5 A8  06
1231 : 48 8A 20 92 11 68 85 A8  D4
1239 : A0 03 B1 AE 99 84 12 88  B4
1241 : 10 F8 60 C9 E4 F0 05 C9  9C
1249 : 2C F0 01 60 20 73 00 20  18
1251 : 2F 0F 8E 84 12 8D 85 12  04
1259 : 20 79 00 C9 A4 F0 05 C9  E9
1261 : 2C F0 01 60 20 73 00 20  30
1269 : 2F 0F 8E 86 12 8D 87 12  64
1271 : CD 85 12 F0 05 B0 0B 4C  3E
1279 : DC 0B EC 84 12 B0 03 4C  F2
1281 : DC 0B 60 00 A0 F8 FF 20  0D
1289 : B0 12 AD 00 DC 29 03 AA  26
1291 : BD 4A 14 8D 20 D0 BD 4E  46
1299 : 14 8D 18 D4 A9 00 18 69  E2
12A1 : 00 A8 88 10 FD AD 01 DC  24
12A9 : C9 FF F0 DE 4C 42 13 20  ED
12B1 : 79 00 C9 E5 D0 10 20 73  4E
12B9 : 00 20 9E B7 8E C1 13 3E  C8
12C1 : 9E 12 CA 8E 24 14 2C 55  2B
12C9 : 15 30 21 AD 11 D0 8D 54  68
12D1 : 14 AD 20 D0 8D 55 14 AD  0D
12D9 : 15 D0 8D 53 14 2C 52 14  38
12E1 : 30 0A A9 00 8D 15 D0 A9  93
12E9 : 0B 8D 11 D0 A9 00 A0 14  5E
12F1 : 99 00 D4 88 10 FA A9 03  56
12F9 : 8D FA FF A9 13 8D FB FF  C6
1301 : 78 60 78 A9 4C 8D 18 14  B6
1309 : 8D BD 13 A9 42 8D 19 14  8C
1311 : 8D BE 13 A9 13 8D 1A 14  26
1319 : 8D BF 13 A9 37 85 01 8D  3F
1321 : 55 15 24 9D 10 0E A9 36  42
1329 : A0 13 20 1E AB A6 AA A5  05
1331 : AB 20 F1 0E 40 53 54 4F  B9
1339 : 50 50 45 44 20 41 54 20  29
1341 : 00 2C 55 15 30 12 AD 53  40
1349 : 14 8D 15 D0 AD 54 14 8D  6C
1351 : 11 D0 AD 55 14 8D 20 D0  B0
1359 : A9 37 85 01 58 A9 8D 8D  43
1361 : 18 14 8D BD 13 A9 18 8D  98
1369 : 19 14 8D BE 13 A9 D4 8D  B4
1371 : 1A 14 8D BF 13 60 A6 37  2E
1379 : A5 38 20 0E 12 20 B0 12  0D
1381 : AD 84 12 85 AA AD 85 12  F8
1389 : 85 AB AD 86 12 8D D1 13  1B
1391 : AD 87 12 8D D7 13 A0 00  D1
1399 : A9 34 85 01 B1 AA 85 A6  B2
13A1 : A9 35 85 01 A0 04 A5 A6  74
13A9 : 0A 69 00 0A 69 00 85 A6  A3
13B1 : 29 03 AA BD 4A 14 8D 20  7A
13B9 : D0 BD 4E 14 8D 18 D4 A2  B0
13C1 : 00 CA 10 FD 88 D0 DF E6  46
13C9 : AA D0 02 E6 AB A5 AA C9  5F
13D1 : 00 D0 C5 A5 AB C9 00 D0  0A
13D9 : BF 4C 42 13 A6 37 A5 38  DC
13E1 : 20 0E 12 20 B0 12 AE C1  6B
13E9 : 13 D0 03 4C DB 17 AD 84  E5
13F1 : 12 85 AA AD 85 12 85 AB  7C
13F9 : AD 86 12 8D 3E 14 AD 87  6A
1401 : 12 8D 44 14 A0 04 AD 00  4E
1409 : DC 29 03 85 A6 AA BD 4A  37
1411 : 14 8D 20 D0 BD 4E 14 8D  C8
1419 : 18 D4 A5 A8 0A 0A 05 A6  6C
1421 : 85 A8 A2 FF CA 10 FD 88  D9
1429 : D0 DC A2 34 86 01 31 AA  A3
1431 : A2 35 86 01 E6 AA D0 02  3B
1439 : E6 AB A5 AA C9 00 D0 C4  1D
1441 : A5 AB C9 00 D0 BE 4C 42  E7
1449 : 13 0E 06 02 0A 07 05 03  18
1451 : 01 00 00 16 FE A9 03 85  0A
1459 : A6 20 9E B7 8A A6 A6 9D  62
1461 : 4E 14 C6 A6 30 0B 20 79  0F
1469 : 00 F0 06 20 FD AE 4C 5A  A2
1471 : 14 60 A9 4A 8D 61 14 A9  F0
1479 : 14 8D 62 14 20 56 14 A9  C7
1481 : 4E 8D 61 14 A9 14 8D 62  A7
1489 : 14 60 80 A9 80 8D 8B 14  ED
1491 : 60 A9 00 8D 8B 14 60 20  93
1499 : 9F 14 8D 52 14 60 C9 91  7F
14A1 : D0 06 20 73 00 A9 80 60  FB
14A9 : C9 E6 D0 06 20 73 00 A9  CB
14B1 : 00 60 20 9E B7 8A C9 00  B4
14B9 : D0 01 60 A9 80 60 20 9E  20
14C1 : AD 20 A3 B6 A5 7A 8D 33  03
14C9 : 15 A5 7B 8D 34 15 86 7A  3C
14D1 : 84 7B 20 C7 12 A9 80 8D  9F
14D9 : 55 15 20 79 00 C9 3A D0  C9
14E1 : 06 20 73 00 4C DB 14 A0  09
14E9 : 05 D9 21 15 F0 15 88 10  C0
14F1 : F8 AD 34 15 85 7B AD 33  C1
14F9 : 15 85 7A A9 00 8D 55 15  91
1501 : 4C 42 13 98 0A A8 B9 27  61
1509 : 15 8D 17 15 B9 28 15 8D  9A
1511 : 18 15 20 73 00 20 E2 FC  B1
```

Commodore

```
1519 : 2C 55 15 10 D4 4C DB 14   7E
1521 : 50 53 57 56 43 23 3E 15   2C
1529 : 47 15 86 10 56 14 73 14   9A
1531 : 35 15 00 00 20 9E AD 20   DF
1539 : A3 B6 4C CF 14 20 28 12   4B
1541 : 20 ED 12 4C 81 13 20 3E   D4
1549 : B7 8E C1 13 8E 9E 12 CA   D6
1551 : 8E 24 14 60 00 A6 37 A5   60
1559 : 38 20 0E 12 AD 84 12 85   BA
1561 : AA AD 85 12 85 AB A9 5F   A1
1569 : 20 D2 FF A6 AA A5 AB 20   8E
1571 : F1 0E A9 06 20 67 11 A9   69
1579 : 3A 20 D2 FF A9 12 20 D2   C9
1581 : FF AD 86 02 48 A9 08 85   36
1589 : A6 78 A2 34 86 01 A0 00   8D
1591 : B1 AA A2 37 86 01 58 20   39
1599 : CF 15 18 E6 AA D0 02 E6   DD
15A1 : AB A5 AB CD 87 12 90 11   31
15A9 : F0 05 68 8D 86 02 60 A5   2D
15B1 : AA CD 86 12 D0 03 4C AB   D3
15B9 : 15 C6 A6 D0 CC 68 8D 86   48
15C1 : 02 A9 0D 20 D2 FF 20 2C   E5
15C9 : A8 F0 DF 4C 67 15 20 DB   C2
15D1 : 15 20 DB 15 20 DB 15 4C   5E
15D9 : DB 15 0A 69 00 0A 69 00   E5
15E1 : 48 29 03 AA BD 4A 14 8D   6E
15E9 : 86 02 8A 49 03 69 31 20   BD
15F1 : D2 FF 68 60 20 2F 0F 86   AE
15F9 : AA 85 AB A9 03 85 A6 20   BD
1601 : 79 00 F0 05 A2 0B 4C 37   79
1609 : A4 20 73 00 F0 45 38 E9   88
1611 : 31 90 F1 C9 05 B0 ED 49   60
1619 : 03 48 A4 A6 B9 57 16 85   F8
1621 : AF 68 A4 A6 F0 06 0A 0A   7E
1629 : 88 4C 25 16 85 AE 78 A9   E6
1631 : 34 85 01 A0 00 B1 AA 25   FF
1639 : AF 05 AE 91 AA A9 37 85   28
1641 : 01 58 C6 A6 10 C3 E6 AA   05
1649 : D0 02 E6 AB A9 03 85 A6   5F
1651 : 4C 0A 16 4C 83 A4 FC F3   EB
1659 : CF 3F 00 A0 F8 FF 00 A0   AD
1661 : F8 FF 00 A0 F8 FF 00 A0   3E
1669 : F8 FF 00 A0 F8 FF 00 A0   46
1671 : F8 FF 00 A0 F8 FF 00 A0   4E
1679 : F8 FF 00 A0 F8 FF 00 A0   56
1681 : F8 FF 00 A0 F8 FF 00 A0   5E
1689 : F8 FF 00 A0 F8 FF 00 A0   66
1691 : F8 FF 00 A0 F8 FF 00 A0   6E
1699 : F8 FF 00 A0 F8 FF 00 A0   76
16A1 : F8 FF 00 A0 F8 FF 00 A0   7E
16A9 : F8 FF 00 A0 F8 FF 00 A0   86
16B1 : F8 FF 00 A0 F8 FF 00 A0   8E
16B9 : F8 FF 00 A0 F8 FF 00 A0   96
16C1 : F8 FF 00 A0 F8 FF 00 A0   9E
16C9 : F8 FF 00 A0 F8 FF 00 A0   A6
16D1 : F8 FF 6F 0C EF 0C 5B 16   1F
16D9 : DB 17 57 4F 52 4B 42 4C   21
16E1 : 4B 2E 20 20 20 20 20 20   13
16E9 : 20 20 20 20 20 20 20 20   E9
16F1 : 20 20 20 20 20 20 20 20   F1
16F9 : 20 20 20 20 20 20 20 20   F9
1701 : 20 20 20 20 20 20 20 20   01
1709 : 20 20 20 20 20 20 20 20   09
1711 : 20 20 20 20 20 20 20 20   11
1719 : 20 20 20 20 20 20 20 20   19
1721 : 20 20 20 20 20 20 20 20   21
1729 : 20 20 20 20 20 20 20 20   29
1731 : 20 20 20 20 20 20 20 20   31
1739 : 20 20 20 20 20 20 20 20   39
1741 : 20 20 20 20 20 20 20 20   41
1749 : 20 20 20 20 20 20 20 20   49
1751 : 20 20 20 20 20 20 20 20   51
1759 : 20 20 20 20 20 20 20 20   59
1761 : 20 20 20 20 20 20 20 20   61
1769 : 20 20 20 20 20 20 20 20   69
1771 : 20 20 20 20 20 20 20 20   71
1779 : 20 20 20 20 20 20 20 20   79
1781 : 20 20 20 20 20 20 20 20   81
1789 : 20 20 20 20 20 20 20 20   89
1791 : 20 20 20 20 20 20 20 20   91
1799 : 20 20 20 20 20 20 20 20   99
17A1 : 20 20 20 20 20 20 20 20   A1
17A9 : 20 20 20 20 20 20 20 20   A9
17B1 : 20 20 20 20 20 20 20 20   B1
17B9 : 20 20 20 20 20 20 20 20   B9
17C1 : 20 20 20 20 20 20 20 20   C1
17C9 : 20 20 46 2D 54 41 53 54   76
17D1 : 45 4E 42 4C 4F 43 4B 54   3C
17D9 : 41 42 20 42 13 4C DC 0B   A9
```

Digitizer Demo Programları

```
DEGISIK HIZLARDA GERI VERME

110 BLOAD"B.DEMO",8 BLOCK31
120 :
130 REM BLOK TABLOSUNDAN SONRA
140 REM SES BILGILERINI YUKLEMEK
150 :
160 BLOAD"S.DEMO",8
170 :
180 REM SESI NORMAL OLARAK
190 REM GERI VERME
200 :
210 PLAY BLOCK 1 SPEED 10
220 :
230 REM IKINCI BLOGU DEGISIK
240 REM HIZLARLA GERI VERMEK
250 :
260 FOR I=1TO5
270 QQ=INT(RND(TI)*8)+6
280 PLAY BLOCK 2 SPEED QQ
290 NEXT I
300 :
310 PLAY BLOCK 3 SPEED 10
READY.
```

Commodore

```
EKO DENEME PROGRAMI

100 REM EKO DENEME PROGRAMI
110 :
120 REM BLOK TABLOSUNU YUKLEMEK
130 :
140 BLOAD"B.DEMO",8 BLOCK 31
150 :
160 REM SES BILGILERINI YUKLEMEK
170 :
180 BLOAD"S.DEMO",8
190 :
200 REM ORNEKLERI GERI VERME
210 :
220 VOLDEF 0,5,10,15
230 :
240 PLAY BLOCK 1 SPEED 10
250 PLAY BLOCK 2 SPEED 10
260 PLAY BLOCK 3 SPEED 10
270 :
280 REM YALNIZCA UCUNCU BLOGU
290 REM TEKRARLE OLARAK GERI
300 REM VERMEK. BU ARADA SES
310 REM SIDDETINI DUZENLI
320 REM OLARAK AZALTMAK.
330 :
340 FOR I=5 TO 0 STEP-.5
350 VOLDEF 0,I,2*I,3*I
360 PLAY BLOCK 3 SPEED 10
370 NEXT I
READY.
```

```
SES BOLUMLERININ TEKRARLANMASI

100 REM TEKRARALI OLARAK
110 REM GERI VERME
120 :
130 REM ONCE BLOK TABLOSU YUKLENIR
140 :
150 BLOAD"B.DEMO",8 BLOCK 31
160 :
170 REM DAHA SONRA SES BILGILERI
180 REM YUKLENIR
190 :
200 BLOAD"S.DEMO",8
210 :
220 REM BLOK TABLOSUNU KULLANARAK
230 REM ORNEKLERI GERI VERME
240 :
250 PLAY BLOCK 1 SPEED 10
260 PLAY BLOCK 2 SPEED 10
270 PLAY BLOCK 2 SPEED 10
280 PLAY BLOCK 3 SPEED 10
READY.
```

Elmaslar

TUNA ERTEMALP

Makine dili ile BASIC'i beraberce kullanan güzel bir oyun. Tam refleksi güçlü olanlara göre.

Fazla açıklama yapmaya gerek yok. Oyun zaten kendini açıklıyor. Programı ilk çalıştırdığınızda, makine dili ile ilgili kısım, 29 saniye içinde hafızaya yerleştiriliyor. Siz bunu ekrandaki saniye sayacı yardımıyla izleyebiliyorsunuz.

Makine dili program hazır olduğu anda oyun başlayacak. Önce size bir açılış ekranı gösterilecek. Bu ekranda oyunu oynamanız için bazı küçük hatırlatmalar var. Bir tuşa basar basmaz da karşınıza iki soru çıkacak. Bu sorulara cevap vererek oyunun zorluk derecesini ayarlayabilirsiniz.

Oyunda neler olacağını anlatırsak, bu sorulara vereceğiniz cevapları anlamanız daha rahat olur. Oyun sırasında, tavana asılı 6 sıra elmas var. Bunlar canları istediği zaman belli bir hızla aşağı doğru düşmeye başlıyorlar. Önce en alt sıranın elmasları düşüyor. Sizin elinizde de altı katlı bir dolap var. Bu dolabı 'L' ve ';' tuşlarının yardımıyla sağa sola kaydırarak elmasları yere düşmeden yakalıyorsunuz. İlk elmas sırası bitince, sizin de dolabınızın katlarından biri eksiliyor. Böylece o ekrandaki elmasların hiçbiri yere düşmeden yakalamaya çalışıyorsunuz. Eğer bir ekrandaki tüm elmas sıraları tükenirse, yeni bir altılı sırayla oyuna devam ediyorsunuz. Ta ki, üç elmas elinizden kurtuluncaya kadar. O zaman oyun bitiyor ve yeni bir oyun oynama hakkınız doğuyor. Ta ki bıkana kadar...

İşte en başta belirttiğimiz iki soru ile, oyunun hızını ve dolabınızın yatay genişliğini saptayabiliyorsunuz. Her ikisine de 1-9 arası değerler verebilirsiniz. Eğer çok acemi olduğunuzu hissediyorsanız, en geniş dolapla en yavaş hızı seçin. Ancak kısa bir süre sonra bundan sıkılacağınıza bahse gireriz. Belki de intihar etmek istersiniz. O zaman da, en dar dolaplarla en yüksek hızı seçin. Daha ne olduğunun farkına bile varmadan oyunu kaybedeceksiniz.

Oyunumuzla neşeli vakitler geçirmenizi dileriz... □

```
PROGRAM ADI : ELMASLAR

00005 POKE 53280,12:POKE 53281,0                          (161)
00007 IF PEEK(49152)<>120 THEN GOSUB 43000
      0                                                   (104)
00009 SYS  49745                                          (068)
00010 PRINT "<CLR><CTRL-2>"TAB(15)"ELMASL
      AR                                                  (085)
00020 PRINT "<5DOWN><CTRL-8><5SPACE>ELMAS
      LARI, YERE DEGMEDEN ONCE                            (238)
00030 PRINT "<DOWN><5SPACE>YAKALAMALISINI
      Z. TAM 5 SANSINIZ                                   (037)
00040 PRINT "<DOWN><5SPACE>VAR.                           (033)
00045 PRINT "<2DOWN><CTRL-2><13SPACE>L -
      SOLA HAREKET                                        (216)
00046 PRINT "<13SPACE>; - SAGA HAREKET                    (250)
00050 PRINT "<5DOWN><C=6><3SPACE><RVSON>L
      UTFEN BIR TUSA BASIN                                (211)
00060 GET A$:IF A$="" THEN 60                             (069)
00065 GOSUB 1000                                          (021)
00070 PRINT "<CLR><CTRL-2>SKOR<2SPACE>000
      00<4SPACE>SANSLAR: QQQQ "                           (045)
00071 SP=53241                                            (093)
00072 PA=12*4096+4095                                     (252)
00073 FL=12*4096+4094:POKE FL,0                           (133)
00074 WI=12*4096+15*256+15*16+11                          (221)
00075 POKE PA,6:POKE WI,W:POKE SP,10-S                    (148)
00078 RO(6)=81:RO(5)=81:RO(4)=207:RO(3)=2
      07:RO(2)=90:RO(1)=90                                (244)
00080 PRINT " <CTRL-8><RVSON)";:FOR I=1 T
      O 38:PRINT "Z";:NEXT :PRINT "<RVSOF
      F> ";                                               (210)
```

Commodore

```
00085 PRINT " <CTRL-8><RVSON>";:FOR I=1 T
      O 38:PRINT "Z";:NEXT :PRINT "<RVSOF
      F> ";                                    <215>
00090 PRINT " <CTRL-4><RVSON>";:FOR I=1 T
      O 38:PRINT "P";:NEXT :PRINT "<RVSOF
      F> ";                                    <156>
00095 PRINT " <CTRL-4><RVSON>";:FOR I=1 T
      O 38:PRINT "P";:NEXT :PRINT "<RVSOF
      F> ";                                    <161>
00100 PRINT " <RVSOFF><C=7>";:FOR I=1 TO
      38:PRINT "W";:NEXT :PRINT " ";           <198>
00102 PRINT " <RVSOFF><C=7>";:FOR I=1 TO
      38:PRINT "W";:NEXT :PRINT " ";           <200>
00105 PRINT "<CTRL-2>";                        <066>
00109 REM                                      <171>
00110 FOR I=1 TO 17:PRINT "<40SPACE>";:NE
      XT                                       <074>
00120 PRINT "<HOME>";                          <050>
00130 FOR I=1984 TO 2023:POKE I,248:POKE
      I+54272,10:NEXT                          <017>
00140 IF  PEEK(783)<>12*16 THEN SYS 12*40
      96                                       <071>
00150 FOR  RO=6 TO 1 STEP -1:FOR  CH=1 TO
       38                                      <155>
00155 FOR K=1 TO 600-CH*10+(6-RO)*20-50*(
      9-PEEK(SP)):NEXT                         <230>
00157 IF PEEK(FL) THEN 2000                    <033>
00160 P=RND(1)*38+1                            <112>
00170 IF PEEK(1024+RO*40+P)=32 THEN 160        <107>
00180 POKE 1024+RO*40+P,RO(RO)                 <070>
00190 NEXT                                     <200>
00191 SYS  49745                               <250>
00192 FOR Q=1 TO 2:POKE 54296,05:POKE 542
      77,5:POKE 54278,218                      <103>
00193 POKE 54273,150:POKE 54272,139:POKE
      54276,17                                 <184>
00194 FOR T=1 TO 50:NEXT :POKE 54276,16:F
      OR T=1 TO 10:NEXT                        <107>
00195 NEXT Q                                   <087>
00197 IF  RO >1 THEN SYS  49691                <034>
00200 NEXT
00201 FOR K=1 TO 300:NEXT K                    <124>
00205 POKE  PA,6                               <211>
00206 IF PEEK(SP)=2 AND PEEK(WI)>1 THEN P
      OKE WI,PEEK(WI)-1                        <125>
00207 IF PEEK(SP)>2 THEN POKE SP,PEEK(SP)
      -1                                       <249>
00210 PRINT "<HOME><DOWN>";                    <193>
00220 GOTO  80                                 <198>
00999 END                                      <239>
01000 PRINT "<CLR><7SPACE>ZORLUK SAYISI <
      5DOWN>"                                  <206>
01010 INPUT "<CTRL-2>HIZI<2SPACE>(1-9)<CT
      RL-8><3RIGHT>5<3LEFT>";S                 <197>
01015 IF S>9 OR S<1 THEN 1010                  <077>
01020 INPUT "<3DOWN><CTRL-2>GENISLIK<2SPA
      CE>(1-9)<CTRL-8><3RIGHT>4<3LEFT>";W      <233>
01030 IF W>9 OR W<1 THEN 1020                  <160>
01040 RETURN                                   <082>
02000 PRINT "<HOME><10DOWN><2SPACE><CTRL-
      8>OYUN BITTI - BOSLUK TUSUNA BASIN       <186>
02010 POKE 198,0                               <140>
02020 GET A$:IF A$<>" " THEN 2020              <059>
02030 RUN 65                                   <042>
49000 PRINT "<CTRL-2><CLR><2DOWN>MAKINA L
      ILI KISMI YERLESIYOR...<3DOWN>":TI$
      ="000000"                                <208>
49005 I=49152                                  <040>
49007 PRINT "KALAN SURE"STR$(29-VAL(TI$))
      " SANIYE <UP>"                           <004>
49010 READ A:IF A=256  THEN  RETURN            <111>
49020 POKE I,A:I=I+1:GOTO 49007                <031>
49152 DATA 120,169,192,141,21,3,169            <019>
49160 DATA 29,141,20,3,88,169,18               <076>
49168 DATA 141,253,207,169,0,141,250           <104>
49176 DATA 207,141,247,207,141,248,207         <026>
49184 DATA 96,173,255,207,141,252,207          <196>
49192 DATA 172,253,207,169,32,153,151          <015>
49200 DATA 7,200,169,160,174,251,207           <097>
49208 DATA 153,151,7,200,202,208,249           <018>
49216 DATA 169,32,153,151,7,206,252            <017>
49224 DATA 207,208,3,76,3,193,172              <146>
49232 DATA 253,207,169,32,153,71,7             <000>
49240 DATA 200,169,160,174,251,207,153         <030>
49248 DATA 71,7,200,202,208,249,169            <002>
49256 DATA 32,153,71,7,200,206,252             <049>
49264 DATA 207,208,3,76,3,193,172              <136>
49272 DATA 253,207,169,32,153,247,6            <054>
49280 DATA 200,169,160,174,251,207,153         <070>
49288 DATA 247,6,200,202,208,249,169           <230>
49296 DATA 32,153,247,6,200,206,252            <184>
49304 DATA 207,240,123,172,253,207,169         <216>
49312 DATA 32,153,167,6,200,169,160            <062>
49320 DATA 174,251,207,153,167,6,200           <085>
49328 DATA 202,208,249,169,32,153,167          <029>
49336 DATA 6,200,206,252,207,240,91            <043>
49344 DATA 172,253,207,169,32,153,87           <192>
49352 DATA 6,200,169,160,174,251,207           <241>
49360 DATA 153,87,6,200,202,208,249            <027>
49368 DATA 169,32,153,87,6,200,206             <024>
49376 DATA 252,207,240,59,172,253,207          <173>
49384 DATA 169,32,153,7,6,200,169              <191>
49392 DATA 160,174,251,207,153,7,6             <035>
49400 DATA 200,202,208,249,169,32,153          <127>
49408 DATA 7,6,200,206,252,207,240             <047>
49416 DATA 27,172,253,207,169,32,153           <092>
49424 DATA 183,5,200,169,160,174,251           <061>
49432 DATA 207,153,183,5,200,202,208           <055>
49440 DATA 249,169,32,153,183,5,200            <143>
49448 DATA 165,197,201,42,208,13,173           <231>
49456 DATA 253,207,201,1,240,24,206            <080>
49464 DATA 253,207,76,40,193,201,50            <088>
49472 DATA 208,14,173,253,207,24,109           <127>
49480 DATA 251,207,201,39,240,3,238            <143>
49488 DATA 253,207,238,250,207,173,250         <037>
49496 DATA 207,205,249,207,240,3,76            <046>
49504 DATA 49,234,169,0,141,250,207            <033>
49512 DATA 169,112,133,251,169,7,133           <045>
49520 DATA 252,160,0,185,152,7,41              <022>
49528 DATA 127,201,32,208,74,200,192           <034>
49536 DATA 39,208,242,160,0,177,251            <083>
49544 DATA 201,81,240,37,201,207,240           <068>
49552 DATA 33,201,90,240,29,200,192            <188>
49560 DATA 40,208,237,56,165,251,233           <056>
49568 DATA 40,133,251,176,2,198,252            <051>
49576 DATA 166,251,208,220,166,252,224         <176>
49584 DATA 4,208,214,76,49,234,170             <044>
49592 DATA 152,24,105,40,168,138,145           <091>
49600 DATA 251,152,56,233,40,168,169           <153>
49608 DATA 32,145,251,32,251,193,76            <169>
49616 DATA 93,193,169,32,153,152,7             <010>
49624 DATA 32,81,194,169,15,141,24             <204>
49632 DATA 212,169,17,141,5,212,169            <103>
49640 DATA 213,141,6,212,169,2,141             <250>
49648 DATA 3,212,169,100,141,2,212             <106>
49656 DATA 169,5,141,1,212,169,135             <204>
49664 DATA 141,0,212,169,65,141,4              <154>
49672 DATA 212,160,0,162,0,142,32              <161>
49680 DATA 208,232,208,250,200,208,247         <117>
49688 DATA 169,12,141,32,208,169,64            <087>
49696 DATA 141,4,212,160,39,185,0              <203>
49704 DATA 4,201,81,240,11,136,208             <029>
49712 DATA 246,169,1,141,254,207,76            <148>
49720 DATA 49,234,169,32,153,0,4               <143>
49728 DATA 76,49,234,152,72,160,10             <183>
49736 DATA 185,0,4,201,57,208,9                <129>
49744 DATA 169,48,153,0,4,136,76               <171>
49752 DATA 255,193,185,0,4,24,105              <114>
49760 DATA 1,153,0,4,104,168,96                <210>
49768 DATA 174,255,207,202,142,255,207         <209>
49776 DATA 232,169,152,133,251,169,7           <212>
49784 DATA 133,252,56,165,251,233,80           <057>
49792 DATA 133,251,176,2,198,252,202           <008>
49800 DATA 208,242,160,0,177,251,201           <032>
49808 DATA 160,240,4,200,76,59,194             <219>
49816 DATA 174,251,207,169,32,145,251          <069>
49824 DATA 200,202,208,250,96,160,0            <153>
49832 DATA 152,153,0,212,200,192,9             <247>
49840 DATA 208,248,96,256                      <039>
```

Joystick Cursor

TUNA ERTEMALP

Amiga gibi aletlerde, ekrandaki bir ok işareti, maus adı verilen bir araç yardımıyla hareket ettirilir. Bu oku bir tür cursor sayabiliriz. Peki Commodore 64 için de böyle bir şey yapılamaz mı? Bizim bir cursor'umuz zaten var. Maus yerine de joystick'i kullansak?

Şimdi programı MDE ile yazıp bir yerlere kaydedin. Sonra da normal LOAD ile yükleyip RUN ile çalıştırın. Görünürde bir şey olmayacak. Şimdi joystick'inizi PORT 2'ye takın ve kolu hareket ettirin. Cursor'un, kolun gösterdiği yönde hareket ettiğini göreceksiniz. RUN-STOP/RESTORE yapmadığınız sürece bu devam edecektir. Eğer kazayla bu tuşlara basarsanız, SYS 49152 ile yeniden joystick-cursor işbirliğini aktif hale getirebiiirsiniz.

Şimdi gelelim teknik ayrıntılara... Programı RUN ile çalıştırdığınızda, esas program kısmı $C000-$C066 (49152-4 9254) arasına yerleşmektedir. Dolayısıyla buralara hiç dokunmamanız lazım.

Programın assembler kodu dökümünü, SUPER-ASS formatında veriyoruz. Birazdan onun üzerinde incelemelerimize başlayacağız.

Joystick'in hareketinin algılanması ve cursor'un hareket ettirilmesi, interrupt programlaması tekniği ile gerçekleştiriliyor. Şimdi bunu biraz açalım...

C64'ün içinde bazı işlemlerin devamlı olarak yapılması gerekmektedir. Örneğin sistem saatinin ilerletilmesi, klavyenin taranması, cursor'un yanıp sönmesi gibi. Ve tüm bu işlemler, bilgisayarın o anda çalıştırdığı programdan bağımsız olmak zorundadır. Bu nedenle, C64'ün sisteminde, interrupt rutini adı verilen bir program parçası vardır. Saniyede 60 kere C64, yaptığı işi bir süre dondurup bu interrupt rutinini işletir. Demin belirttiğimiz birtakım işler bitince, daha önce işleyen ana program kaldığı yerden çalışmaya devam eder. İnterrupt rutininin toplam çalışma süresi 1/60 saniyeden çok çok daha kısa olduğu için, saniyede 60 kere çağırılması zaman açısından bir sorun yaratmamaktadır. Bu rutin, sistemin içinde $EA31 (59953) adresinden başlamaktadır. Bu adres, $EA ve $31 olarak büyük ve küçük byte'larına ayrılabilir. İşte bu iki sayı da, $0314/$0315 (788/799) adreslerinde saklanır.

Biz tam burada işe karışıyoruz. $0314/$0315'e kendi yazdığımız bir rutinin başlangıç adresini yazıyoruz. Artık bilgisayar saniyede 60 kere bizim rutinimizi çalıştıracaktır. Ancak dikkat edilmesi gerekli bir iki küçük nokta var. Öncelikle rutinimiz uzun olmamalıdır. Eğer çok uzun olursa, zaman harcaması artacak ve interrupt rutininin çalışması yüzünden esas işlemlere zaman kalmayacaktır. Bir de, eğer kendi interrupt işlemlerinizi kendiniz yazacak kadar profesyonel programcı değilseniz, yazdığınız minik interrupt programınızın en sonuna JMP $EA31 koyarak esas interrupt rutinini çağırın.

Şimdi gelelim bizim programa... SYS 49152 yaptığınızda, 200-230 arasında gördüğünüz gibi interrupt vektörü $C00F adresine değiştiriliyor. Bu adres, assembler kodunun 280 numaralı satırına denk geliyor. 240-250 arasında da bir sayaç tanımlanıyor. Saniyede 60 kere cursor'u oynatmak çok hızlı olacağı için, saniyede 10 kere işlem yapılacak. Bunun için, $02 numaralı adres sayaç olarak kullanılacak. Burası 6'dan 0'a geri sayılacak ve 0 olduğunda joystick sorulacak. Sonra bu işlem tekrarlanacak.

280-300 arasında interrupt rutininin ilk yaptığı işlemleri görüyorsunuz. $02'deki sayaç bir azaltılıyor ve sıfır olup olmadığı kontrol ediliyor. Eğer sıfır değilse $EA31'deki normal interrupt rutinine atlanıyor. Aksi durumda ise, öncelikle bizim programın joystick ile ilgili kısmı çağırılıyor.

320'de ilk yapılan iş, sayacı yeniden 6 ile yüklemek. Sonra da C64'ün sisteminin, basılan ve klavye deposuna gönderilen tuşların sayısını sakladığı $C6 (198) adresi okunuyor. Eğer klavye deposu (keyboard buffer) dolmuşsa, doğrudan $EA31'e atlanıyor.

Dolmamışsa, işlem 390'dan devam ediyor. Önce akümülatör PORT 2'nin gönderdiği değer ile yükleniyor. 0 numaralı bit kontrol ediliyor.

Commodore

Buradan çıkan sonuç, joystick'in yukarı doğru hareketi ise, akümülatör CRSR UP kodu olan $91 (145) ile yükleniyor ve rutinin çıkış noktasına atlanıyor.

Eğer hareket yukarı değilse, işlem 450'den devam ediyor. Önce akümülatör PORT 2'nin gönderdiği değer ile yükleniyor. 1 numaralı bit kontrol ediliyor. Buradan çıkan sonuç, joystick'in aşağı doğru hareketi ise, akümülatör CRSR DOWN kodu olan $11 (17) ile yükleniyor ve rutinin çıkış noktasına atlanıyor.

Eğer hareket aşağı değilse, işlem 510'dan devam ediyor. Önce akümülatör, PORT 2'nin gönderdiği değer ile yükleniyor. 2 numaralı bit kontrol ediliyor. Buradan çıkan sonuç, joystick'in sola doğru hareketi ise, akümülatör CRSR LEFT kodu olan $9D (157) ile yükleniyor ve rutinin çıkış noktasına atlanıyor.

Eğer hareket sola değilse, işlem 570'den devam ediyor. Önce akümülatör PORT 2'nin gönderdiği değer ile yükleniyor. 3 numaralı bit kontrol ediliyor. Buradan çıkan sonuç, joystick'in sağa doğru hareketi ise, akümülatör CRSR RIGHT kodu olan $1D (29) ile yükleniyor ve rutinin çıkış noktasına atlanıyor.

Eğer joystick bu hareketlerin hiçbirini yapmamışsa veya rutinin çıkış noktasındaki işlemler tamamlanmışsa, program 630'da $EA31'e atlıyor. Ancak bir kol hareketi varsa, demin de gördüğümüz gibi akümülatörün içine yüklenen cursor kontrol karakteri 650'den sonraki kısımda klavye deposuna gönderiliyor ve geri dönülerek joystick'in okunmasına devam ediliyor.

Aynı mantıkla birçok değişik program yazabileceğinizi, bazı kısa programlarınızı arka planda yürütebileceğinizi belirterek yazımızı bitirelim...

64'ER'den Uyarlanmıştır

```
 PROGRAM  : JOY-CURSOR          0801 0894
-----------------------------------------

0801 : 0B 08 C1 07 9E 32 30 36  0A
0809 : 31 00 00 00 A9 2C A0 08  C9
0811 : 85 5F 84 60 A9 34 A0 08  45
0819 : 85 5A 84 5E A9 68 A0 C0  3A
0821 : 85 58 84 59 20 BF A3 4C  46
0829 : 00 C0 00 A9 0F 8D 14 03  72
0831 : A9 C0 8D 15 03 A9 06 85  E1
0839 : 02 60 C6 02 F0 03 4C 31  18
0841 : EA A9 06 85 02 A6 C6 E0  64
0849 : 08 90 03 4C 31 EA AD 00  05
0851 : DC 29 01 C9 00 D0 05 A9  29
0859 : 91 20 5E C0 AD 00 DC 29  48
0861 : 02 C9 00 D0 05 A9 11 20  84
0869 : 5E C0 AD 00 DC 29 04 C9  4D
0871 : 00 D0 05 A9 9D 20 5E C0  26
0879 : AD 00 DC 29 08 C9 00 D0  F3
0881 : 05 A9 1D 20 5E C0 4C 31  26
0889 : EA A6 C6 9D 77 02 E6 C6  DC
0891 : 60 00 C0 85 60 20 8E B4  7D

READY.
```

```
80   -;          JOYSTICK CURSOR
90   -;
100  -           .EQ EL02=$02
110  -           .EQ ELC6=$C6
120  -           .EQ EL0277=$0277
130  -           .EQ EL0314=$0314
140  -           .EQ EL0315=$0315
150  -           .EQ ELDC00=$DC00
160  -           .EQ ELEA31=$EA31
170  -;
180  -           .BA $C000
190  -;
200  -           LDA #$0F        ; "."
210  -           STA EL0314
220  -           LDA #$C0        ; "-"
230  -           STA EL0315
240  -           LDA #$06        ; "."
250  -           STA EL02
260  -           RTS
270  -;
280  -           DEC EL02
290  -           BEQ LC016
300  -           JMP ELEA31
310  -;
320  -LC016      LDA #$06        ; "."
330  -           STA EL02
340  -           LDX ELC6
350  -           CPX #$08        ; "."
360  -           BCC LC023
370  -           JMP ELEA31
380  -;
390  -LC023      LDA ELDC00
400  -           AND #$01        ; "."
410  -           CMP #$00        ; "."
420  -           BNE LC031
430  -           LDA #$91        ; "."
440  -           JSR LC05E
450  -LC031      LDA ELDC00
460  -           AND #$02        ; "."
470  -           CMP #$00        ; "."
480  -           BNE LC03F
490  -           LDA #$11        ; "."
500  -           JSR LC05E
510  -LC03F      LDA ELDC00
520  -           AND #$04        ; "."
530  -           CMP #$00        ; "."
540  -           BNE LC04D
550  -           LDA #$9D        ; "."
560  -           JSR LC05E
570  -LC04D      LDA ELDC00
580  -           AND #$08        ; "."
590  -           CMP #$00        ; "."
600  -           BNE LC05B
610  -           LDA #$1D        ; "."
620  -           JSR LC05E
630  -LC05B      JMP ELEA31
640  -;
650  -LC05E      LDX ELC6
660  -           STA EL0277,X
670  -           INC ELC6
680  -           RTS

READY.
```

Koca Harfler

TUNA ERTEMALP

Programlarında, bildiğimiz küçük harflerin yanısıra, koca koca harfler de kullanmak isteyenler varsa, bu programa göz atsınlar.

Bu program sayesinde, BASIC programınızda kullanacağınız bir SYS komutu ile, istediğiniz yazıyı ekranın istediğiniz yerinde normalden 16 kat daha büyütülmüş olarak yazmanız mümkün.

Önce MDE kullanarak programı yazınız ve SAVE ediniz. ,8,1 veya ,1,1 kullanarak disketten veya kasetten yükleyin ve hemen NEW verin. Sonra da DEMO programını yazın. Çalıştırdığınızda, programımızın özelliklerini göreceksiniz.

Rutini çağıran SYS komutu şu şekilde:

SYS 53000,satır,kolon,yazı

Örneğin SYS 53000,10,11,"deneme" verdiğinizde, "deneme" yazısı ekranın 10. satırında 11. kolondan başlamak üzere büyük boy karakterlerle yazılacaktır. "deneme" yerine başka şeyler de koyabilirsiniz. Örneğin, A$ veya 2+4 gibi. Bu gibi durumlarda, "yazı" yerine verdiğiniz terim bilgisayar tarafından işlenecek ve ekrana yazılacak hale getirilecektir.

Satır için 0-21 arası, kolon içinse 0-26 arası sayılar belirtilebilir. Yazının ekrana yazılması, yazacak yer kalmayıncaya kadar veya yazının sonu gelinceye kadar devam edecektir. Eğer yazınızın içinde renk değiştirme, ekran silme gibi kontrol karakterleri varsa, bunlar normal işlevlerini görecektir. Böylece, büyük boy harfle yazılarınızın içinde renkten renge geçmeniz çok kolay. Hattâ RVS ON ve RVS OFF tuşlarını bile kullanıp, negatif karakterleri de büyük boyda yazdırabilirsiniz.

BASIC'deki demo programını biraz inceleyelim. Önce çeşitli renk kontrol karakterleri bir karakter değişkenine yükleniyor. Bunu daha sonra yazımızın rengini değiştirmekte kullanacağız. Sonra ekranı silip T sayacını açılış durumuna getiriyoruz. Artık döngümüze başlayabiliriz. Döngümüz ekran satırlarını 0-21 arasında dörder dörder tarıyor. T ile de, kaçıncı satırı yazdığımızı sayıyoruz. Eğer T'nin içindeki sayı tek sayı ise, o satır negatif yazılıyor. Eğer çift ise, negatif olmayan karakterler kullanılıyor.

15. satırda, gerektiği kadar C-64 yazısı yazılıyor. 30. satır ise, sırası gelen satır ve kolona, büyük boy C-64 yazısını yazıyor. Sonraki iki satır, 15-20 arası satırların yaptığını, ekranın sol tarafı için de tekrarlamaktadır. 100. satırda da döngümüz bitiyor.

Meraklı arkadaşlarımız için, büyük boy karakterlerinin nasıl oluştuğunu inceleyebilmeleri amacıyla, programın SUPER-ASS formatında hazırlanmış assembler kodunu da veriyoruz.

Artık programlarınızı şenlendirmenin vakti geldi... □

```
 PROGRAM  : KOCA  HARFLER    CF08 D000
-----------------------------------

CF08 : A9 2F 85 00 A9 37 85 01   17
CF10 : 20 F1 B7 E0 16 B0 21 86   AB
CF18 : 03 20 F1 B7 E0 25 B0 18   C9
CF20 : 86 04 20 FD AE 20 9E AD   32
CF28 : 24 0D 30 06 20 DD BD 20   C8
CF30 : 87 B4 20 A6 B6 AA D0 03   F8
CF38 : 4C 48 B2 86 05 A0 00 78   6C
CF40 : B1 22 AA 0A C9 40 8A B0   18
CF48 : 06 20 16 E7 4C E6 CF 84   25
CF50 : 60 0A B0 0C C9 C0 90 03   4E
CF58 : 29 BE 2C 29 7E 4C 68 CF   9C
CF60 : C9 FE D0 02 A9 BC 09 80   C2
CF68 : 85 5C AD 18 D0 29 02 09   FA
CF70 : 34 A6 C7 F0 02 09 01 06   80
CF78 : 5C 2A 06 5C 2A 85 5D A6   88
```

Commodore

```
CF80 : 03 18 BD F0 EC 65 04 85   32
CF88 : 58 85 5A B5 D9 29 03 6D   BE
CF90 : 88 02 85 59 29 03 09 D8   26
CF98 : 85 5B A0 00 A2 00 A9 33   2A
CFA0 : 85 01 A1 5C 85 5E E6 5C   39
CFA8 : A1 5C 85 5F E6 5C A9 37   2B
CFB0 : 85 01 A9 00 06 5E 2A 06   28
CFB8 : 5E 2A 06 5F 2A 06 5F 2A   3E
CFC0 : AA BD F0 CF 91 58 AD 86   1E
CFC8 : 02 91 5A C8 98 29 03 D0   C3
CFD0 : E1 18 98 69 24 A8 C0 A0   DC
CFD8 : 90 C2 A5 04 69 03 85 04   80
CFE0 : C9 25 B0 0A A4 60 C8 C4   A3
CFE8 : 05 B0 03 4C 3F CF 58 60   24
CFF0 : 20 6C 7B 62 7C E1 FF FE   46
CFF8 : 7E 7F 61 FC E2 FB EC A0   31

READY.
```

```
80    -;              KOCA HARFLER
90    -;
100   -               .EQ EL00=$00
110   -               .EQ EL01=$01
120   -               .EQ EL03=$03
130   -               .EQ EL04=$04
140   -               .EQ EL05=$05
150   -               .EQ EL0D=$0D
160   -               .EQ EL22=$22
170   -               .EQ EL58=$58
180   -               .EQ EL59=$59
190   -               .EQ EL5A=$5A
200   -               .EQ EL5B=$5B
210   -               .EQ EL5C=$5C
220   -               .EQ EL5D=$5D
230   -               .EQ EL5E=$5E
240   -               .EQ EL5F=$5F
250   -               .EQ EL60=$60
260   -               .EQ ELC7=$C7
270   -               .EQ ELD9=$D9
280   -               .EQ EL0286=$0286
290   -               .EQ EL0288=$0288
300   -               .EQ ELAD9E=$AD9E
310   -               .EQ ELAEFD=$AEFD
320   -               .EQ ELB248=$B248
330   -               .EQ ELB487=$B487
340   -               .EQ ELB6A6=$B6A6
350   -               .EQ ELB7F1=$B7F1
360   -               .EQ ELBDDD=$BDDD
370   -               .EQ ELD018=$D018
380   -               .EQ ELE716=$E716
390   -               .EQ ELECF0=$ECF0
400   -;
410   -               .BA $CF08
420   -;
430   -               LDA #$2F        ; "/"
440   -               STA EL00
450   -               LDA #$37        ; "7"
460   -               STA EL01
470   -               JSR ELB7F1
480   -               CPX #$16        ; "."
490   -               BCS LCF38
500   -               STX EL03
510   -               JSR ELB7F1
520   -               CPX #$25        ; "%"
530   -               BCS LCF38
540   -               STX EL04
550   -               JSR ELAEFD
560   -               JSR ELAD9E
570   -               BIT EL0D
580   -               BMI LCF32
590   -               JSR ELBDDD
600   -               JSR ELB487
610   -LCF32          JSR ELB6A6
620   -               TAX
630   -               BNE LCF3B
640   -LCF38          JMP ELB248
650   -;
660   -LCF3B          STX EL05
670   -               LDY #$00        ; "."
680   -LCF3F          SEI
690   -               LDA (EL22),Y
700   -               TAX
710   -               ASL
720   -               CMP #$40        ; "@"
730   -               TXA
740   -               BCS LCF4F
750   -               JSR ELE716
760   -               JMP LCFE6
770   -;
780   -LCF4F          STY EL60
790   -               ASL
800   -               BCS LCF60
810   -               CMP #$C0        ; "_"
820   -               BCC LCF5B
830   -               AND #$BE        ; "■"
840   -               .BY $2C         ; ","
850   -LCF5B          AND #$7E        ; "."
860   -               JMP LCF68
870   -;
880   -LCF60          CMP #$FE        ; "."
890   -               BNE LCF66
900   -               LDA #$BC        ; "■"
910   -LCF66          ORA #$80        ; "."
920   -LCF68          STA EL5C
930   -               LDA ELD018
940   -               AND #$02        ; "."
950   -               ORA #$34        ; "4"
960   -               LDX ELC7
970   -               BEQ LCF77
980   -               ORA #$01        ; "."
990   -LCF77          ASL EL5C
1000  -               ROL
1010  -               ASL EL5C
1020  -               ROL
1030  -               STA EL5D
1040  -               LDX EL03
1050  -               CLC
1060  -               LDA ELECF0,X
1070  -               ADC EL04
1080  -               STA EL58
1090  -               STA EL5A
1100  -               LDA ELD9,X
1110  -               AND #$03        ; "."
```

Commodore

```
1120 -           ADC EL0286
1130 -           STA EL59
1140 -           AND #$03          ; "."
1150 -           ORA #$08          ; "♣"
1160 -           STA EL5B
1170 -           LDY #$00          ; "."
1180 -LCF9C      LDX #$00          ; "."
1190 -           LDA #$33          ; "3"
1200 -           STA EL01
1210 -           LDA (EL5C,X)
1220 -           STA EL5E
1230 -           INC EL5C
1240 -           LDA (EL5C,X)
1250 -           STA EL5F
1260 -           INC EL5C
1270 -           LDA #$37          ; "7"
1280 -           STA EL01
1290 -LCFB2      LDA #$00          ; "."
1300 -           ASL EL5E
1310 -           ROL
1320 -           ASL EL5E
1330 -           ROL
1340 -           ASL EL5F
1350 -           ROL
1360 -           ASL EL5F
1370 -           ROL
1380 -           TAX
1390 -           LDA TLCFF0,X
1400 -           STA (EL58),Y
1410 -           LDA EL0286
1420 -           STA (EL5A),Y
1430 -           INY
1440 -           TYA
1450 -           AND #$03          ; "."
1460 -           BNE LCFB2
1470 -           CLC
1480 -           TYA
1490 -           ADC #$24          ; "$"
1500 -           TAY
1510 -           CPY #$A0          ; " "
1520 -           BCC LCF9C
1530 -           LDA EL04
1540 -           ADC #$03          ; "."
1550 -           STA EL04
1560 -           CMP #$25          ; "%"
1570 -           BCS LCFEE
1580 -           LDY EL60
1590 -LCFE6      INY
1600 -           CPY EL05
1610 -           BCS LCFEE
1620 -           JMP LCF3F
1630 -;
1640 -LCFEE      CLI
1650 -           RTS
1660 -;
1670 -TLCFF0     .BY $20,$6C,$7B,$62,$7C,$E1,$FF,$FE; " ......."
1680 -           .BY $7E,$7F,$61,$FC,$E2,$FB,$EC; "......."
1690 -TLCFFF     .BY $A0           ; " "

READY.
```

Koca Harfler Demo

```
00001 CL$="<CTRL-1><C=8><CTRL-5><C=7><C=1
      ><C=3>"                                    <207>
00003 PRINT "<CLR>"                              <247>
00005 T=-1                                       <203>
00010 FOR I=0 TO 21 STEP 4:T=T+1                 <040>
00011 IF T/2<>INT(T/2) THEN PRINT "<RVSON
      >";:GOTO 15                                <224>
00012 PRINT "<RVSOFF>";                          <190>
00015 POKE 214,I:POKE 211,0:SYS 58640:PRI
      NT MID$(CL$,T+1,1);                        <000>
00020 IF I<>0 THEN FOR I1=1 TO T:PRINT "C
      -64";:NEXT                                 <193>
00030 SYS 53000,I,T*4,"C-64"                     <082>
00035 POKE 214,I:POKE 211,16+4*T:SYS 5864
      0:PRINT MID$(CL$,T+1,1);                   <028>
00040 FOR I1=1 TO 6-T:PRINT "C-64";:NEXT         <099>
00100 NEXT                                       <110>
```

Commodore

Sentetik Melodi

TUNA ERTEMALP

Şu kısacık programı yazıp çalıştırınca kulaklarınıza inanamayacaksınız!!! Sizlere dünyanın en kısa müzik programını sunuyoruz. Üstelik sizin çalmanıza gerek bile yok. Kendi bestesini kendisi yapıyor.

Size verdiğimiz minik programı yazıp çalıştırın. Bilgisayarınızın ilk bestesini duyacaksınız. Herhangi bir tuşa basın. İşte ikinci beste. Canınız sıkılana kadar devam edin. Sıkılınca da, RUN/STOP tuşu ile programı durdurun.

Önce BASIC programına bir göz atalım.

Onuncu satır, program için bir hazırlık. Otuzuncu satırda makine dilinde yazılmış kısa program, 832-890 ($0340-$037A) adresleri arasına yerleştiriliyor. Otuzbeşinci satırın ne yaptığı belli. Otuzaltıncı satır ise, müzik besteliyor! Programın geri kalanı, kullanılan makine dili programını DATA satırları halinde saklıyor.

832'den başlayıp SYS 832 ile çağırılan makine dili programı, 1022 ve 1023 adresleri içinde bulunan değerlere göre ses üretiyor. Kullanılan algoritma nedeniyle de, ortaya oldukça ilginç sonuçlar çıkıyor. Esas önemli olan, bu iki sayı arasındaki fark. Bu nedenle birisine sıfır koyup diğerinde 0-255 arası sayıları denemek de mümkün. Bizim programımız her ikisini de RND fonksiyonu ile saptayıp, olayı tamamen tesadüfi hale getiriyor. İsterseniz kendi bestenizi yazabilirsiniz. Bu arada belirtelim, birazdan da göreceğiniz gibi, tuşa basmanızı bekleyen program, BASIC programı değil, makine dili programı. Kendi programınızı yazarken buna dikkat edin. Eğer makine dili biliyorsanız, verdiğimiz assembler kodu yardımıyla ve tabiî ki SUPER-ASS'ı kullanarak, istediğiniz değişiklikleri yapın.

Madem söz açıldı, şimdi de makine dilindeki esas programı, assembler kodu üzerinden açıklamaya girişelim... Bu işi bilenler okusunlar, bilmeyenlere şimdiden iyi besteler dileriz...

100-170 satırları arasında, programda kullanılan dış adresler label'lara yükleniyor. Programın başlangıç adresinin $0340 (832) olduğunu 190. satırda belirtiyoruz. Programın ilk yaptığı şey, ses şiddetini en yüksek seviyeye çıkarmak. Aynı zamanda sesin şiddetini kaybetme hızı en yavaş düzeye indirilirken, düşeceği seviye 7 olarak belirleniyor. Bu iki işlem, aynı anda sesin yükselme hızını maksimum yapıp, tınıyı olduğu gibi ortadan kaldırıyor. Eğer bu terimleri tam bilemiyorsanız, C-64 Kullanıcı El Kitabı'nda ve Programcının El Kitabı'nda bulabilirsiniz.

260-270 arasında, üçgen dalga biçimi seçiliyor. 280 numaralı satırdan itibaren esas besteleme kısmı başlıyor. 1023 adresinin içeriği bir artırılırken, 1022'nin içeriği bir azaltılıyor. 300-360 arası bir tür bekleme döngüsü. 370'de bir tuşa basılıp basılmadığı kontrol ediliyor. Eğer basılmamışsa, program 390'dan devam ediyor. 1022 ve 1023'ün içerikleri AND işlemine sokuluyor ve sonuç ses frekansının üst byte'ı olarak kullanılıyor. Ve program tekrar başa dönüyor.

Ancak, bir tuşa basıldığında program 430'dan devam ediyor. Ses şiddeti ve klavyede basılan tuşları sayan adres ($C6=198) sıfırlanıyor. Sonra da, BASIC programına geri dönülüyor.

İşte bu kadar. Güzel, değil mi? □

```
PROGRAM ADI : SENTETIK MELODI
00010 POKE 1022,0                                          <012>
00030 FOR T=832 TO 890:READ Q:POKE T,Q:NE
      XT                                                   <198>
00035 PRINT "<CLR>BIR TUSA BASIN"                          <065>
00036 POKE 1023,RND(8)*256:POKE 1022,RND(
      8)*256:SYS 832:GOTO 36                               <027>
00041 DATA 169,15,141,24,212,141,5,212,16
      9,112,141,6,212,169,17,141,4,212,23
      8                                                    <181>
00042 DATA 255,3,206,254,3,162,0,16,0,200
```

```
      ,208,253,232,224,7,208,246,165,198  (064)
00043 DATA 208,11,173,255,3,45,254,3,141,
      1,212,176,223,169,0,133,198,141,24  (182)
00044 DATA 212,96                         (001)

READY.

80   -;          SENTETIK MELODI
90   -;
100  -           .EQ ELC6=$C6
110  -           .EQ EL03FE=$03FE
120  -           .EQ EL03FF=$03FF
130  -           .EQ ELD401=$D401
140  -           .EQ ELD404=$D404
150  -           .EQ ELD405=$D405
160  -           .EQ ELD406=$D406
170  -           .EQ ELD418=$D418
180  -;
190  -           .BA $0340
200  -;
210  -           LDA #$0F        ; "."
220  -           STA ELD418
230  -           STA ELD405
240  -           LDA #$70        ; "."
250  -           STA ELD406
260  -           LDA #$11        ; "."
270  -           STA ELD404
280  -L0352      INC EL03FF
290  -           DEC EL03FE
300  -           LDX #$00        ; "."
310  -L035A      BPL L035C
320  -L035C      INY
330  -           BNE L035C
340  -           INX
350  -           CPX #$07        ; "."
360  -           BNE L035A
370  -           LDA ELC6
380  -           BNE L0373
390  -           LDA EL03FF
400  -           AND EL03FE
410  -           STA ELD401
420  -           BCS L0352
430  -L0373      LDA #$00        ; "."
440  -           STA ELC6
450  -           STA ELD418
460  -           RTS

READY.
```

Renkli List

TUNA ERTEMALP

BASIC programlarınızın dökümlerini renklendirmek ister misiniz?

BASIC'in en kötü yanlarından biri, PASCAL gibi modüler programlamaya kolaylık sağlamamasıdır. Siz gene de altprogramları kullanma yoluyla bayağı modüler bir program yazabilirsiniz. Ama sağolsun BASIC'in LIST komutu, hiçbir modülü görüntü olarak belli edemeyecektir. Görüntüyü de modüler yapabilmek için, REM'lerle, boşluklarla dolu bir program yaratmanız gerekecektir. Boşuna hafıza kaybı...

Peki, programınızın her modülü LIST sırasında farklı renklerde görünse nasıl olur?

Yapacağınız şey, her modülünüzün (altprogramınızın) başına bir REM satırı koymak ve REM komutunun arkasına en az bir karakterden oluşan bir yazı yazmak. BASIC programınızı böylece hazırladıktan sonra SAVE edin.

Sonra da, RENKLİ LIST programını yazıp çalıştırın. Şimdi de, demin hazırlıklarını yaptığınız modüler programınızı LOAD edin ve LIST komutunu verin. Her REM komutundan sonra gelen kısmın farklı bir renk ile LIST edildiğini göreceksiniz.

İşte bu rengi, REM komutundan sonra gelen ilk karakter belirleyecektir. Eğer arka arkaya gelen iki modülün aynı renge sahip olmasını istiyorsanız, REM'den sonra aynı harfi koyun. Eğer ayrı harfler koyarsanız, renkler de farklı olacaktır. Hoşunuza giden renkleri deneme yanılma metodu ile bulabilirsiniz. Örneğin A harfi beyaz, B harfi kırmızı döküm yaratacaktır.

Şimdi işin biraz teknik ayrıntısına girelim. Eğer bir REM komutunun o ana kadar geçerli olan rengi değiştirmesini istiyorsanız, REM ile yazacağınız harf arasında boşluk OLMAMALIDIR. Dolayısıyla, o ana kadar kullanılan rengin herhangi bir REM satırından sonra da korunmasını istiyorsanız, REM ile arkasından gelen yazı arasına en az bir boşluk koymalısınız. Bu arada dikkat edin, bir satırda sadece REM varsa, dökümün geri kalan kısmı siyah olacaktır.

Eğer RUN/STOP/RESTORE yaparsanız, renkli döküm işlemi bitecektir. Tekrar başlatmak için POKE 774,0: POKE 775,192 vermeniz gerekiyor.

Peki bu renkli döküm işi nasıl oluyor? Gördüğünüz gibi, programımız kısa bir makine dili rutinini 49152 ($C000) adresinden sonraya yerleştiriyor. Bir sonraki POKE komutları ile de, normalde sistemdeki BASIC komutlarını ekranda yazıya çeviren rutinin adresini saklayan 774/775 adreslerine, 49152'de başlayan yeni rutinin adresi yerleştiriliyor. Bir anlamda, LIST komutu önce bizim yazdığımız minik rutini işletecek, sonra kendi işini görecek hale getiriliyor. Bu makine dili rutinini bir parça inceleyelim:

```
c000   48       pha
c001   c9 8f    cmp # $8f
c003   d0 0b    bne $c010
c005   c8       iny
c006   b1 5f    lda ($5f),y
c008   c9 20    cmp # $20
c00a   f0 03    beq $c00f
c00c   8d 86 02 sta $0286
c00f   88       dey
c010   68       pla
c011   4c la a7 jmp $a71a
```

Önce akümülatördeki değer yığın içine (stack) atılıyor. Sonra da, o sırada ekrana LIST edilen komutun REM olup olmadığı kontrol ediliyor. REM'in kodu $8F'dir. Eğer öyle değilse, $C010'a atlanıp akümülatörün değeri yenileniyor ve gerçek LIST rutinine atlanıyor.

Ancak REM'e rastlanmışsa, hemen arkasından gelen karakter okunup, eğer boşluk değilse, $0286 (646) numaralı adrese yazılıyor. Bu adres, ekrandaki yazının rengini kontrol eder. Eğer karakter boşlukla veya renk kodu 646'a yerleştirilmişse, Y indeksi başlangıç durumuna getirilip akümülatör yeniden orijinal değeri ile yükleniyor ve sistemdeki gerçek LIST rutinine atlanıyor. Bu atlama sırasında, gerek Y yazmaçı gerekse akümülatör başlangıçtaki değerlerine sahip oldukları için, esas LIST rutininin bu olaylardan haberi olmuyor.

İşte bu kadar. İşinize yaramasını dileriz... □

```
PROGRAM ADI : RENKLI LIST

00010 DATA 72,201,143,208,11,200,177,95,2
      01,32,240,3,141,134,2,136,104,76,26
      ,167                                          <097>
00020 FOR I=49152 TO 49171:READ A:POKE I,
      A:NEXT                                        <144>
00030 POKE 774,0:POKE 775,192                       <065>
```

Commodore

Çizgi Cursor

TUNA ERTEMALP

Profesyonel iş bilgisayarlarının çoğunda cursor tüm karakteri kaplamaz, sadece karakterin altında çizgi olarak görünür. Neden COMMODORE 64'ümüzün de böyle bir cursor'u olmasın?

Küçük programımız bu işi yapıyor. Programı çalıştırdığınızda, bu işin oluşmasını adım adım seyretme imkânı bulacaksınız. Biz burada işin teknik açıklamasını yapalım.

İlk önce yazı karakteri rengi için açık mavi, arka plan rengi için de koyu mavi seçiliyor. İlk satırda gördüğünüz, önce arka plan rengini açık mavi yapıp ekranı silmek sonra da arka plan rengini yeniden değiştirmek, karakterlere ait renk kodlarının saklandığı renk RAM'ını tamamen açık mavinin koduyla doldurmak için kullanılan küçük bir hiledir. Bir FOR... NEXT döngüsü kullanmaktan daha hızlıdır. Ancak bu hile belli bir modelden sonraki COMMODORE 64'lerde değişiklik gösteriyor. Eğer programı gördüğünüz şekilde yazar da bir sonuç alamazsanız, POKE 53281,14 yerine POKE 646,14 yapın. Bu adres, ekrandaki yazı karakterlerinin rengini kontrol eder.

Renklendirme işlemi bitince, ekran hafızası (VideoRAM) 0-255 arası karakter kodları ile doldurulmakta. Böylece COMMODORE 64'ün tüm karakter seti gözlerinizin önüne serilecek. Dikkat ederseniz bir noktaya kadar normal olan karakterler, o noktadan sonra negatifleşmekte. İşte bizim de saldıracağımız nokta burası olacak.

Ekranda cursor olarak gördüğümüz şey, aslında o an cursor'un üstünde bulunduğu karakterin normal ve negatif biçimlerinin arka arkaya gösterilmesinden başka bir şey değil. Bu durumda, eger negatif karakterleri değiştirir ve sadece en alt kısımlarında düz bir çizgi kalacak şekilde yeniden oluşturursak, bilgisayar o karakterin negatifini gösterdiğini sandığı zaman, aslında altı çizili halini gösteriyor olacak. Şimdi bu değişiklikleri nasıl yaptığımıza bakalım...

Önce bazı değişkenleri ayarladıktan sonra, VIC'in 24 numaralı yazmaçı ile oynayarak (bkz. Sayı 10 ve 12, COMMODORE 64 İLE GRAFİK) karakter setinin ve videoram'ın başlangıç adreslerini değiştiriyoruz. Karakter seti 53248'de, videoram'da 52224'de aktif hale gelmiştir artık. Ancak bu işin tam olması için CIA 2'nin (56576) içeriği ile de oynamak gerekiyor. Bu yöntemle BANK 4'ü seçerek her şeyi yerli yerine oturtuyoruz. Bir de 648'de ekran editörüne ekranın yeni yerini bildirince, her şey tamam oluyor.

Artık ekranın ve yeni karakter setinin kopyalarını bu bölgelere çıkarmak lazım. Bu yüzden önce 56333 ve 1 numaralı adreslerin içeriklerini değiştirerek, bilgisayarımızın kesintiye uğramasını önleyip karakter ROM'unu okumayı olanaklı kılıyoruz.

Ve 2 numaralı satırda ekranın içeriğinin taşınması ile işimize başlıyoruz. Ancak bu satırda ekranda sadece saçmalık göreceksiniz, çünkü karakter setini daha tanımlamadık. Programın geri kalan kısmında da, 3. satırın sonuna kadar eski karakter setinin bir kopyasını yeni yerine, üzerinde gerekli değişiklikleri yaparak çıkarıyoruz. İşte bu işlemin seyrine doyum olmuyor, çünkü tüm karakterlerin değişimini ekranda izleyebiliyorsunuz.

Artık işimiz bitmiştir. 4. satırda 1 ve 56333'deki değerleri eski haline döndürüp bilgisayarı normale döndürebiliriz. Cursor'umuzu da uygun bir yere yerleştirince, profesyonel cursor'umuz emrimize hazırdır... □

PROGRAM ADI : CIZGI CURSOR

```
00000 POKE 53281,14:PRINT "<CLR>";:POKE 5
      3281,6:FOR I=0 TO 255:POKE 1024+I,I
      :NEXT                                  <095>
00001 W=56333:Q=53248:Z=4↑5:POKE Q+24,52:
      POKE 56576,0:POKE 648,204:POKE W,1:
      POKE 1,3                               <235>
00002 FOR I=0 TO 999:POKE 52224+I,PEEK(Z+
      I):NEXT :FOR I=0 TO Z:A=Q+I:B=A+2*Z
      :L=Z*((I AND 7)=7)                     <149>
00003 POKE A,PEEK(A):POKE B,PEEK(B):POKE
      A+Z,PEEK(A-L):POKE B+Z,PEEK(B-L):NE
      XT                                     <037>
00004 POKE 1,7:POKE W,129                    <131>
00005 PRINT "<11DOWN>"                       <002>
```

Disketteki Programların Silinmeye Karşı Korunması:

Program Kilidi- SB

SUAT BAYCILI

Mecmuamızın 11. sayısındaki "DİSKET-KURTARAN-SB" isimli programımı okumuş olanlar, SCRATCH komutu ile disketten yanlışlıkla silinen programların nasıl kurtarılabildiğini görmüşlerdir.

Peki, programları silinmeye karşı korumanın bir yolu yok mudur?

Gerçi silinen bir programı artık nasıl kurtarıp ortaya çıkarabileceğimizi biliyoruz, fakat işi baştan sağlam tutup istediğimiz programları silinemez hale getirmek daha yerinde olacaktır. İşte aşağıdaki programımız bu işi başarı ile yapmaktadır.

TRACK: 18/SECTOR: 1

Gerek silinmiş bir programın nasıl kurtarıldığını, gerekse kilitleme işleminin nasıl yapıldığını anlamak için disketteki BANT: 18/SEKTÖR: 1'i yakından incelememiz gerekmektedir.

Bilindiği gibi, 1541 Disk Sürücüsü ile 4040 tipinde formatlanmış bir diskette 35 Bant **(Track)** bulunmaktadır.

Bu Bantlardan 18 no.lu olanı özellikle çok önemlidir. Bu bantta meydana gelecek bir hasar disketin bir kısmının veya tamamının kullanılamaz hale gelmesine neden olabilir.

Diskette bulunan programlarla ilgili bütün veriler (bu programların adları, disket üzerinde nerede başlayıp nerede bittikleri, hangi tip oldukları, silinmiş veya kilitli olup olmadıkları gibi) bu bantta kayıtlıdır.

18. Bantta toplam 19 adet sektör vardır ve her bir sektör 256 bayttan oluşur. İlk, yani 0 no.lu sektörde BAM, Disket Adı, ID no.su, format tipi gibi temel veriler vardır.

Geri kalan 18 adet sektörde ise disketteki programlarla ilgili veriler vardır, şöyle ki:

Bayt		No:	Bulunan Veriler
0	ve	1	Bir sonraki sektörün Blok ve Sektör no.su,
2	ila	31	Directory'deki 1. dosyaya ait veriler,
34	ila	63	Directory'deki 2. dosyaya ait veriler,
66	ila	95	Directory'deki 3. dosyaya ait veriler,
.....		.....	
226	ila	255	Directory'deki 8. dosyaya ait veriler.

Geri kalan sektörlerde de aynı sayıda programa ait veriler kayıtlı olduğundan toplam 18 X 8 = 144 eder, yani bir diskete en fazla 144 adet dosya kaydedilebilir, tabii bunların toplam uzunluğu da en fazla 664 blok olabilir.

Şimdi de örnek olarak 2 ila 31 no.lu baytlar arasında yer alan (Directory'deki 1. programa ait) verilere göz atalım:

Bayt		No:	Bulunan Veriler
0			Dosya tipi
1	ve	2	Bu dosyaya ait ilk veri (data) blokunun Bant ve Sektör no.su
3	ila	18	Dosya adı
19	ila	21	Yalnız Relatif dosyaya ait bazı veriler
22	ila	25	(kullanılmaz)

Commodore

26 ve 27 OPEN@ için, değiştirilen dosyanın Bant ve Sektör no.su.
28 ve 29 Dosyanın blok adedi, Alt Bayt (LB) ve Üst Bayt (HB) olarak.

Diğer dosyalara ait veriler de aynı düzen içinde sıralandığından 2 ila 31 no.lu baytlar 0 ila 29 olarak yazılmıştır.

Şimdi gelelim işin püf noktasına!

Silinmiş bir programın kurtarılması da, bir programın silinmeye karşı kilitlenmesi de, hep dosya tipini belirleyen baytın değiştirilmesi ile sağlanmaktadır. Bant: 18 / Sektör: 1'de bulunabilecek 8 dosya için bu baytlar: 2, 34, 66, 96, 130, 162, 194 ve 226 no.lu baytlardır.

Bu baytların normal değerleri ile dosya silindiği veya kilitlendiği zamanki değerleri şöyledir:

Dosya tipi	Normal değer
SEQ (Sırasal)	129
PRG (Program)	130
USR (Kullanıcı)	131
REL (Relatif)	132

Silinince	Kilitlenince
128	193
128	194
128	195
128	196

Diyelim ki, yeni formatlanmış bir diskete bir Program Dosyası SAVE ettiniz. Bant: 18/ Sektör: 1'de bu dosyaya ait veriler 2 ila 31 no.lu baytlar arasında yer alacaktır. PRG tipi dosya olduğuna göre 2 no.lu baytın değeri 130'dur.

Şimdi, kanal açıp SCRATCH komutu ile bu dosyayı silersek bu baytın değeri 128'e döner ve BAM'da bu dosyaya ait blokların yerleri boş olarak görünmeye başlar, yani gerçekte dosya silinmiş olmaz. DİSKET-KURTARANSB, bu baytın değerini 130'a çevirir, BAM'ı da düzeltir. Böylece, silinmiş bir dosya da kurtarılmış olur.

PROGRAM-KİLİDİ-SB de benzer şekilde çalışmaktadır. 2 no.lu baytın normalde 130 olan değerini 194'e çevirirsek bu dosya kilitlenmiş olur ve artık SCRATCH komutu ile silinemez hale gelir.

Bir dosyanın kilitlenmesinin veya silinmiş bir dosyanın kurtarılmasının prensip yönünden çok basit olduğu görülüyor. Ancak pratikte kurtarmak veya kilitlemek istediğiniz programın dosya tipini gösteren baytı bulup değiştirmeniz o kadar kolay değildir. Eğer elinizde "Disk Doctor" veya "Disk Monitor" gibi programlar varsa anlattığımız işlemleri bunlar yardımı ile kolayca yapabilirsiniz. Ancak bu tür programlar disketlerin Bant ve Sektör düzenlerinin, ASCII kodlarının ve Hekzadesimal sistemin iyi bilinmesini gerektirir. Halbuki, DİSKET-KURTARAN-SB ve PROGRAM-KİLİDİ-SB'yi kullanmak için bunların hiçbirini bilmenize gerek yoktur. Yükleyip RUN edin yeter.

Nasıl Çalışıyor?

Programı RUN ettiğinizde karşınıza önce bir Menü çıkacaktır.

F1 MENÜ
F3 PROGRAM KİLİTLEME (LOCK)
F5 KİLİT AÇMA (UNLOCK)
F7 DİRECTORY
← SON

Buna göre, F1 Menüye geri dönmenizi, F7 Directory'yi görmenizi sağlamaktadır. ←ile ise programa son verip BASIC'e dönebilirsiniz. İçinde kilitlemek istediğiniz programların bulunduğu disketi yerleştirip F3'e basarsanız size kilitleyeceğiniz programın adını soracaktır. Hatırlayamazsanız F7'e basarak Directory'i görebilir veya F1'e basarsanız vazgeçip Menü'ye geri dönebilirsiniz. Programın adını doğru ve tam olarak yazıp RETURN'e basarsanız program kilitlenmiş olur. F7'ye basıp Directory'i listelerseniz bu programın adının hizasında en sağda bir < işareti göreceksiniz. Bu işaret programın kilitlenmiş olduğunu gösteren standart bir semboldür.

Eğer programın silinmeye karşı kilitlendiğine inanmıyorsanız, F1 ile Menü'ye geçip ← ile BASIC'e döndükten sonra (veya doğrudan doğruya bilgisayarı açıp kapadıktan sonra) bu programı silmeyi deneyelim. Programın adı da QWERTY olsun.

OPEN 15, 6, 15, "S: QWERTY" CLOSE 15 yazıp RETURN'e basalım. Normal olarak bu komut QWERTY isimli programı siler, daha doğrusu BANT: 18'de bu programın dosya tipini gösteren baytı 126 yapar. Ancak siz LOAD "$" ve LIST komutları ile Directory'i listelerseniz QWERTY isimli programın hâlâ yerinde olduğunu göreceksiniz. Çünkü artık silinmeye karşı "kilitlenmiş" durumdadır.

Şimdi biz açıklamalarımıza devam edelim. Eğer programın adını doğru yazmazsanız veya *"wild card"* kullanırsanız (yani programın adının ilk birkaç harfini yazıp sonuna koyarsanız) programı kilitleyemezsiniz. Ayrıca disketin yanındaki çentiğin üzeri bantla kapatılmışsa kilitleme işlemi gerçekleşmez ve bir uyarı ibaresi çıkar. Keza kilitlenmiş bir programı tekrar kilitlemek isterseniz "ZATEN KİLİT ALTINDA" ibaresi çıkacaktır.

Kilitli bir programı silmek isterseniz önce kilidi açmanız gerekir.

Bunun için Menü çıktığında F5'e basınız ve kilidini açmak istediğiniz programın adını yazınız. Daha sonra F7'ye basarak Directory'e baktığınızda programın adının hizasında en sağda bulunan işaretinin ortadan kalktığını göreceksiniz, artık bu programı isterseniz silebilirsiniz. Yanlışlıkla silerseniz ne mi olur, DİSKET-KURTARAN-SB programınızı kurtarır.

Bu program (daha doğrusu hiçbir program) yardımı ile disketinizdeki değerli programları FORMATLAMA yolu ile silinmeye karşı koruyamazsınız. Çünkü ID no.su. belirtilerek yapılan formatlama disketin üzerinde ne var ne yoksa siler ve geriye tertemiz bir disket kalır. Bu yüzden, kullanılmış bir disketi formatlamadan önce üzerinde değerli bir program olup olmadığını kontrol etmekte büyük fayda vardır

Diğer bir tehlike kaynağı da, bir disketin tamamının kopyesini çıkaran BACKUP programlarıdır. Bunlar arka arkaya dolu disket ("master" veya "source" disket) ile boş disket ("destination" veya "target" disket) birkaç defa disk sürücüye takıp çıkarmakla dolu disketin tam bir kopyesini boş diskete aktarır. İşte bu sırada yapacağınız bir sıra hatası dolu disketinizin kısmen veya tamamen silinmesine yol açabilir. Çaresi, dolu disketin yanındaki çentiği önceden bantla kapatmaktır.

Program İle İlgili Açıklamalar

Şimdi biraz da BASIC ile yazılmış programımızın nasıl çalıştığını inceleyelim:

Satır 116: Program ilk iş olarak GOSUB 290 ile, 290 ila 390 nol.lu satırlar arasındaki iki makine kodu bölümünü okumaktadır. Bunları sona koymanın amacı, buradaki DATA rakamlarını bellekteki yerlerine POKE etmeye yalnız bir kere ihtiyaç oluşudur.

Satır 290-294: Bellekte 828 ila 1006 arasına 304 ila 346 no.lu satırlardaki DATA'ları yerleştiren bu bölüm, sizin kilitlemek veya kilidini açmak üzere adını yazacağınız dosyanın BANT: 18'de nerede olduğunu bulmakta, dosya tipini belirten baytın yerini P=PEEK (255) ile bulmanızı sağlamakta ve dosyanın durumunu A=PEEK (252) ile görmenizi sağlamaktadır. Bu işleri yapmak için 160 ve 192 no.lu satırlardaki SYS 828 komutu verilmektedir.

Satır 296-298: Bu bölüm ise bellekte 50000 ila 50124 no.lu adreslere, bellekte bulunan programı silmeden Directory okumayı sağlayan ve 360 ila 390 no.lu satırlarda görülen DATA'lardan oluşan makine kodunu POKE etmektedir. Ne zaman F7'ye basılsa SYS 50000 komutu ile ekrana Directory gelmektedir.

Satır 116: RETURN ile dönülen bu satırda, programın çalışırken durdurulmasını önlemek için 788 ve 792 no.lu adreslere 52 ve 193 değerleri yerleştirildikten sonra 244 no.lu satıra atlanmaktadır.

Satır 244-274: Bu bölüm ekrana Menü'yü getirmekte ve basacağınız tuşa göre gereken satıra atlamayı sağlamaktadır.

Satır 150-176: F3'e basmakla geçilen bu bölüm, program kilitleme bölümüdür. Burada önce GOSUB 128 ile ekran düzenlenmektedir. İstediğiniz anda F1 veya F7'ye basarak Menü'ye geri dönmeniz veya Directory'i görmeniz sağlanmaktadır. Daha sonra RETURN ile geri dönüldüğünde, GOSUB 236 ile atlanan bölümde, vermiş olduğunuz dosya adı eğer 16 karakterden kısa ise, SHIFTED SPACE'lerle 16'ya tamamlandıktan sonra bellekte 683 ila 699 no.lu adresler arasına ASCII kodları halinde yerleştirilmektedir. Bu bölüm her dosya adı verildiğinde SYS 828 ile verilen dosyanın diskette bulunup bulunmadığını aramakta kullanılmaktadır.

Satır 214-226: 158 no.lu satırdaki GOSUB 214 ile atlanan bu bölümde,

OPEN 15, 6, 15, "IO": OPEN 2, 8, 2, "#" ile disk sürücü önce initialize edildikten sonra bir rastgele erişim **(random access)** kanalı açılmakta, sonra da hata kanalı okunmaktadır. Bant: 18/Sektör: 1 için Satır 118'de U1 yani Buffer Pointer'i değiştirmeyen bir Block-Read komutu verildikten sonra GOTO 216 ile hata kanalı kontrol edilip Satır 160'a dönülmektedir. SYS 828 ile, verilen dosya adı Bant: 18'de kayıtlı dosyalarla karşılaştırılarak A=PEEK (252) ile dosya tipi tesbit edilmektedir. Eğer (A AND 64)>0 ise bu A'nın değerinin 193 ile 196 arasında olduğunu gösterir ki, dosya zaten kilitli demektir. Aksi halde P=PEEK (255) ile dosya tipini gösteren baytın yeri bulunduktan sonra,

PRINT#15, "B-P" ;2; P:PRINT#2, CHR$ (A OR 64) ile yeri P olan baytın değeri 64 ile OR'lanmaktadır. Kilitli olmayan dosyalara ait 129, 130, 131, 132 değerleri 64 ile OR'lanırsa 193, 194, 195, 196 değerleri elde edilir ki, dosya da kilitlenmiş olur.

Satır 182-208: Bu bölüm bir dosyanın kilidini açmaktadır. Burada ise eğer (A AND 64)>0 ise bu defa A değeri 135 ile AND'lenerek 193 ile 196 arası değerler 129 ilie 132 arası değerlere çevrilmekte yani dosyanın kilidi açılmaktadır. Eğer (A AND 64)=0 ise dosya zaten kilitli değil demektir.

Bu açıklamalardan sonra aşağıda programımızı görüyorsunuz. □

PROGRAM ADI : PROG-KILIDI-SB

```
00100 REM  ************************************                <238>
00102 REM  **      PROGRAM KILIDI-SB          **                <225>
00104 REM  **      ------------------         **                <213>
00106 REM  **         SUAT BAYCILI            **                <051>
00108 REM  ************************************                <246>
00110 :                                                        <086>
00112 PRINT "<CLR><CTRL-7><10DOWN><11SPACE>LUTFEN BEKLEYINIZ."  <165>
00114 F1=4:F7=3:POKE  53281,12:POKE  53280,6                   <233>
00116 GOSUB  290:POKE  788,52:POKE  792,193:GOTO  244          <030>
00118 PRINT#15,"U1";2;0;T;S:GOTO  216                          <243>
00120 PRINT#15,"B-P 2 0":PRINT#15,"U2";2;0;T;S:GOTO  216       <050>
00122 T=PEEK(681):S=PEEK(682):RETURN                           <224>
00124 :                                                        <100>
00126 :                                                        <102>
00128 PRINT :PRINT "<3SPACE><RVSON>F1<RVSOFF> MENU":PRINT "<3SPACE><RVSON
      >F7<RVSOFF> DIRECTORY"                                   <036>
00130 PRINT "<2DOWN><3SPACE>PROGRAM ADI ? <CTRL-7>T<CTRL-1>";:POKE  198,0
      :F$=""                                                   <022>
00132 QQ=PEEK(203):GET  A$:IF  A$=""  AND  QQ=64  THEN  132    <075>
00134 IF  QQ=F1  OR  QQ=F7  THEN  PRINT  CHR$(20):RETURN       <217>
00136 IF  A$=CHR$(20)  AND  F$=""  THEN  132                   <166>
00138 IF  A$=CHR$(13)  AND  F$<>""  THEN  PRINT  CHR$(20):RETURN  <143>
00140 IF  A$=CHR$(13)  AND  F$=""  THEN  132                   <175>
00142 IF  A$=CHR$(20)  THEN  PRINT  CHR$(20)A$"<CTRL-7>T<CTRL-1>";:F$=LEF
      T$(F$,LEN(F$)-1):GOTO  132                               <032>
00144 PRINT  CHR$(20)A$"<CTRL-7>T<CTRL-1>";:F$=F$+A$:GOTO  132  <035>
00146 :                                                        <122>
00148 :                                                        <124>
00150 PRINT "<CLR><DOWN><RVSON><CTRL-3><8SPACE>PROGRAM KILITLEME <LOCK><8
      SPACE><CTRL-1>".                                         <216>
00154 GOSUB  128:IF  QQ=F1  THEN  RETURN                       <111>
00156 IF  QQ=F7  THEN  SYS  50000:GOSUB  278:GOTO  150         <206>
00158 POKE  673,1:GOSUB  236:GOSUB  214:T=18:S=1               <225>
00160 GOSUB  118:SYS  828:A=PEEK(252):IF  A  THEN  166         <042>
00162 GOSUB  122:IF  T  THEN  160                              <056>
00164 PRINT "<DOWN><CTRL-3><3SPACE>PROGRAM BULUNAMADI<CTRL-1><DOWN>":CLOS
      E  2:CLOSE  15:GOTO  154                                 <230>
00166 IF (A  AND  64)  THEN  PRINT "<2DOWN><RVSON>"F$"<RVSOFF>";"<RVSON><
```

```
      CTRL-7> ZATEN KILIT ALTINDA!<3SPACE><RVSOFF>"                          <077>
00168 IF (A AND 64) THEN CLOSE 2:CLOSE 15:GOSUB 278:GOTO 176                 <044>
00170 P=PEEK(255):GOSUB 118:PRINT#15,"B-P";2;P:PRINT#2,CHR$(A OR 64);        <248>
00172 GOSUB 120:IF E THEN 150                                                <235>
00174 CLOSE 2:CLOSE 15:PRINT "<2DOWN><RVSON>"F$"<RVSOFF>";"<RVSON><CTRL
      -3> ARTIK KILITLI!<3SPACE><RVSOFF>":GOSUB 278                          <061>
00176 POKE 673,0:GOTO 150                                                    <057>
00178 :                                                                      <154>
00180 :                                                                      <156>
00182 PRINT "<CLR><DOWN><RVSON><CTRL-3><11SPACE>KILIT ACMA (UNLOCK)<10SPA
      CE><CTRL-1>"                                                           <161>
00186 GOSUB 128:IF QQ=F1 THEN RETURN                                         <143>
00188 IF QQ=F7 THEN SYS 50000:GOSUB 278:GOTO 182                             <114>
00190 POKE 673,1:GOSUB 236:GOSUB 214:T=18:S=1                                <001>
00192 GOSUB 118:SYS 828:A=PEEK(252):IF A THEN 198                            <088>
00194 GOSUB 122:IF T THEN 192                                                <144>
00196 PRINT "<DOWN><CTRL-3><3SPACE>PROGRAM BULUNAMADI<CTRL-1><DOWN>":CLOS
      E 2:CLOSE 15:GOTO 186                                                  <118>
00198 IF (A AND 64)=. THEN PRINT "<2DOWN><RVSON>"F$"<RVSOFF>";"<RVSON
      ><CTRL-7> ZATEN KILITLI DEGIL!<3SPACE>"                                <046>
00200 IF (A AND 64)=. THEN CLOSE 2:CLOSE 15:GOSUB 278:GOTO 208               <041>
00202 P=PEEK(255):GOSUB 118:PRINT#15,"B-P";2;P:PRINT#2,CHR$(A AND 135)
      ;                                                                      <144>
00204 GOSUB 120:IF E THEN 182                                                <067>
00206 CLOSE 2:CLOSE 15:PRINT "<2DOWN><RVSON>"F$"<RVSOFF>";"<RVSON><CTRL
      -3> KILIDI ACILDI!<3SPACE><RVSOFF>":GOSUB 278                          <206>
00208 POKE 673,0:GOTO 182                                                    <145>
00210 :                                                                      <186>
00212 :                                                                      <188>
00214 CLOSE 2:CLOSE 15:OPEN 15,8,15,"I0":OPEN 2,8,2,"#"                      <137>
00216 INPUT#15,E,EM$,ET,ES                                                   <096>
00218 IF E=26 OR E=73 THEN CLOSE 2:CLOSE 15:GOTO 224                         <009>
00220 IF E THEN PRINT "<DOWN><CTRL-3><3SPACE><RVSON>"E"<LEFT>,"EM$","E
      T"<LEFT>,"ES"<RVSOFF><CTRL-1>":CLOSE 2:CLOSE 15:END                    <024>
00222 RETURN                                                                 <024>
00224 IF E=26 THEN PRINT "<2DOWN><CTRL-3><3SPACE>DISKETIN KORUMA BANDIN
      I CIKARINIZ!":GOSUB 278:RETURN                                         <204>
00226 IF E=73 THEN PRINT "<2DOWN><CTRL-3><3SPACE>DISKETIN FORMATI FARK
      LI!":GOSUB 278:RETURN                                                  <242>
00232 :                                                                      <208>
00234 :                                                                      <210>
00236 IF LEN(F$)<16 THEN F$=F$+CHR$(160):GOTO 236                            <039>
00238 FOR I=1 TO LEN(F$):POKE 683+I,ASC(MID$(F$,I,1)):NEXT :POKE 700
      ,0:RETURN                                                              <083>
00240 :                                                                      <216>
00242 :                                                                      <218>
00244 PRINT "<CLR><DOWN><RVSON><CTRL-3><11SPACE>PROGRAM-KILIDI-SB<12SPACE
      >"                                                                     <056>
00246 PRINT "<CTRL-1><2DOWN><5SPACE><RVSON>F1<RVSOFF> MENU"                  <196>
00248 PRINT "<DOWN><5SPACE><RVSON>F3<RVSOFF> PROGRAM KILITLEME (LOCK)"       <234>
00250 PRINT "<DOWN><5SPACE><RVSON>F5<RVSOFF> KILIT ACMA<6SPACE>(UNLOCK)"     <204>
00252 PRINT "<DOWN><5SPACE><RVSON>F7<RVSOFF> DIRECTORY"                      <199>
00254 PRINT "<DOWN><5SPACE><RVSON>←<RVSOFF><2SPACE>SON"                      <035>
00256 PRINT "<CTRL-3><2DOWN><8SPACE>SECIMINIZI YAPINIZ."                     <079>
00258 GET A$:IF A$="" THEN 258                                               <234>
00260 IF A$<>"<F1>" AND A$<>"<F3>" AND A$<>"<F5>" AND A$<>"<F7>"
      AND A$<>"←" THEN 258                                                   <189>
00262 IF A$="<F1>" THEN 244                                                  <063>
00264 IF A$="<F3>" THEN GOSUB 150:GOTO 244                                   <234>
00266 IF A$="<F5>" THEN GOSUB 182:GOTO 244                                   <220>
00268 IF A$="<F7>" THEN GOSUB 272:GOTO 244                                   <087>
00270 IF A$="←" THEN 274                                                     <076>
00272 PRINT "<CTRL-1>":SYS 50000:GOSUB 278:RETURN                            <082>
```

Commodore

```
00274 PRINT "<CTRL-1>":CLOSE  2:CLOSE  15:POKE  788,49:POKE  732,71:END   <084>
00276 :                                                                   <254>
00278 PRINT "<2DOWN><CTRL-3><3SPACE>DEVAM ETMEK ICIN <RVSON>SPACE<RVSOFF>
      'E BASINIZ.<CTRL-1>"                                                <053>
00280 GET  A$:IF  A$=""  THEN  280                                        <189>
00282 IF  A$<>" "  THEN  280                                              <134>
00284 PRINT "<CLR>":RETURN                                                <231>
00286 :                                                                   <008>
00288 :                                                                   <117>
00290 POKE  679,0:POKE  140,0                                             <082>
00292 S=0:FOR  I=828  TO  1006:READ  A:POKE  I,A:S=S+A:NEXT               <116>
00294 IF  S<>26645  THEN  PRINT "<CTRL-2><2DOWN><10SPACE>1.GRUP DATA'DA H
      ATA!":END                                                           <042>
00296 S=0:FOR  I=50000  TO  50124:READ  A:POKE  I,A:S=S+A:NEXT            <235>
00298 IF  S<>18638  THEN  PRINT "<CTRL-2><2DOWN><10SPACE>2.GRUP DATA'DA H
      ATA!":END                                                           <251>
00300 RETURN                                                              <104>
00302 :                                                                   <024>
00304 DATA  169,0,141,168,2,133,253,162                                   <241>
00306 DATA  2,32,198,255,32,228,255,141                                   <020>
00308 DATA  169,2,32,228,255,141,170,2                                    <114>
00310 DATA  230,253,230,253,32,228,255,133                                <143>
00312 DATA  252,165,253,133,255,32,228,255                                <015>
00314 DATA  32,228,255,230,253,230,253,160                                <076>
00316 DATA  0,32,228,255,230,253,153,189                                  <046>
00318 DATA  2,200,24,192,16,144,242,160                                   <165>
00320 DATA  243,32,228,255,230,253,165,144                                <077>
00322 DATA  240,3,141,168,2,200,208,241                                   <144>
00324 DATA  165,252,208,7,173,168,2,208                                   <003>
00326 DATA  22,240,191,165,140,240,3,32                                   <151>
00328 DATA  198,3,173,167,2,240,3,32                                      <106>
00330 DATA  175,3,173,168,2,240,171,32                                    <246>
00332 DATA  204,255,96,160,0,185,172,2                                    <018>
00334 DATA  240,8,217,189,2,208,7,200                                     <080>
00336 DATA  208,243,238,168,2,96,169,0                                    <047>
00338 DATA  133,252,96,160,0,185,189,2                                    <056>
00340 DATA  240,6,32,210,255,200,208,245                                  <111>
00342 DATA  165,252,41,64,240,10,169,144                                  <184>
00344 DATA  32,210,255,169,60,32,210,255                                  <132>
00346 DATA  169,13,32,210,255,169,144,32                                  <069>
00348 DATA  210,255,96                                                    <181>
00350 :                                                                   <072>
00360 DATA  169,48,133,252,169,2,133,253                                  <141>
00362 DATA  76,95,195,169,1,133,253,169,0                                 <052>
00364 DATA  133,144,169,36,133,251,169,251                                <122>
00366 DATA  133,187,169,0,133,188,165,253                                 <209>
00368 DATA  133,183,169,8,133,186,169,96                                  <118>
00370 DATA  133,185,32,213,243,165,186,32                                 <240>
00372 DATA  180,255,165,185,32,150,255,164                                <035>
00374 DATA  144,208,61,160,6,132,251,32                                   <018>
00376 DATA  165,255,166,252,133,252,164                                   <173>
00378 DATA  144,208,46,164,251,136,208,238                                <246>
00380 DATA  164,252,32,205,189,169,32,32                                  <131>
00382 DATA  210,255,32,165,255,166,144,208                                <039>
00384 DATA  24,170,240,6,32,210,255,76,170                                <008>
00386 DATA  135,169,13,32,210,255,165,137                                 <025>
00388 DATA  201,63,240,4,160,4,208,137,32                                 <055>
00390 DATA  66,246,96                                                     <248>
```

C 28 Püf Noktalarının Programları

PROGRAM 1

```
100 BA=INPUT("D600")
110 INPUT"VIDEO RAM ADRESI";V
120 INPUT"DEGER";W
130 HO=INT(V/256):LO=V-(256*HI)
140 POKE BA,18:POKE BA+1,HI
150 POKE BA,19:POKE BA+1,LO
160 POKE BA,31:POKE BA+1,W
170 WAIT BA,32
180 POKE BA,30:POKE BA+1,1
```

PROGRAM 2

```
10 HO=KOL+PEEK(238)*SR
20 S1=PEEK(224):S2=PEEK(225)
30 HI=(AD/256):LO=AD-(256*HI)
40 POKE244,LO:POKE225,HI
50 SYS49155,KRTR,RENK
60 POKE224,S1:POKE225,S2
70 END
```

PROGRAM 3

```
10 FOR X=6144 TO 6182
20 READ A:CS=CS+A:POKE X,A
30 NEXT X
40 IF CS<>3411 THEN PRINT CHR$(7);:LIST
```

```
50 DATA  72,138, 72,152, 72,169,  2,141, 40, 10,162, 18
60 DATA 104, 32, 27, 24,232,104, 32, 27, 24,162, 31,104
70 DATA  76, 27, 24,142,  0,214, 44,  0,214, 16,251,141
80 DATA   1,214,96
```

PROGRAM 4

```
20 BA=8192
30 A=INPUT("D600"):B=A+1
40 FOR X=0 TO 7
50 AD=BA+X+8*W: IF AD>16383 THEN END
60 HI=INT(AD/256):LO=AD-(256*HI)
70 POKE A,18:POKE B,HI
80 POKE A,19:POKE B,LO
90 POKE A,31:CH=PEEK(B)
100 FOR Y=7 TO 0 STEP -1
110 IF CH>=2↑Y THEN CH=CH-2↑Y:PRINT "*";:GOTO PRINT ".";
120 NEXT Y
130 PRINT
140 NEXT X
150 W=W+1
160 GOTO 40
```

PROGRAM 5

```
10 INPUT"KARKETER";Z$ :GOSUB 60000:PRINT Z$;:PRINT"[illegible]":  GOTO 10
60000 Z=ASC(Z$):Z$="":IF Z AND 128 THEN X=Z AND 127 OR 64:GOTO 60040
60010 IF NOT Z AND 64 THEN X=Z:GOTO 60040
60020 IF Z AND 32 THEN X=Z AND 95:GOTO 60040
60030 X=Z AND 63
60040 L$="":U$=""
60050 FOR W=0 TO 7
60060 L$=L$+CHR$(157)
60070 U$=U$+CHR$(145)
60080 NEXT W
60090 D$=CHR$(17)
60100 REM KAREKTER URETICI SECIMI
60110 A=INPUT("D600"):B=A+1
60130 FOR Z=0 TO 7:REM 8 BYTE
60140 AD=8192+X*16+Z
60150 HI=INT(AD/256):LO=AD-(256*HI)
60170 POKE A,18:POKE B,HI
60180 POKE A,19:POKE B,LO
60190 POKE A,31:C=PEEK(B)
60210 FOR Y=7 TO 0 STEP -1 :REM 8 PIKSEL SATIR/KAREKTER
60220 IF C>=2↑Y THEN C=C-2↑Y:ZE$=ZE$+"*":GOTO ZE$=ZE$+" "
60230 NEXT Y
60240 Z$=Z$+ZE$+L$+D$:ZE$=""
60250 NEXT Z
60260 Z$=LEFT$(Z$,LEN(Z$)-9)+U$
60270 RETURN
READY.
```